**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**




**Grado en Ingeniería Informática**

# TRABAJO FIN DE GRADO

## APLICACIÓN WEB PARA EL SEGUIMIENTO DEL CICLO DE VIDA DE PERSONAS CON TRASTORNO DEL ESPECTRO AUTISTA


**Jhunnior Javier Jimbo Calva**
**Tutor: Germán Montoro Manrique**


**Junio de 2019**

# APLICACIÓN WEB PARA EL SEGUIMIENTO DEL CICLO DE VIDA DE PERSONAS CON TRASTORNO DEL ESPECTRO AUTISTA

**AUTOR: Jhunnior Javier Jimbo Calva**
**TUTOR: Germán Montoro Manrique**

**Dpto. Ingeniería Informática**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Junio de 2019**

# Resumen

Este Trabajo Fin de Grado tiene como objetivo el desarrollo de una aplicación web para realizar el seguimiento del ciclo de vida de las personas con trastornos del espectro autista, sustituyendo así la forma en la que se realiza actualmente dicho seguimiento; la aplicación está destinada para los alumnos de la asociación Alenta.

Para realizar la implementación web se ha utilizado el *framework* Django, además de otras herramientas como el motor de base de datos PostgreSQL y el *front-end framework* Bootstrap.

La aplicación denominada **Camino de Vida**, por una parte, permite el registro de los usuarios que van a realizar un seguimiento de su ciclo de vida. En dicho registro se crea un perfil de usuario que contiene la información de los alumnos, así como imágenes de distintas etapas de su desarrollo. Por otra parte, tenemos el componente característico de esta aplicación, el seguimiento de su ciclo de vida, para llevarlo a cabo existe una serie de tareas, en donde se almacenan las valoraciones que los alumnos hacen respecto a su habilidad para realizar las tareas que deben ir aprendiendo en su ciclo de vida. A partir de estas tareas evaluadas se genera un camino de vida con una serie de metas, en donde los alumnos pueden visualizar su progreso en el ciclo de vida.

De acuerdo a esto, el proyecto de Django ha sido modulado en 2 aplicaciones: *Accounts* y *Tasks*, que respectivamente se encargan de las partes que acabamos de mencionar.

Destacar que para el diseño de la aplicación se ha seguido la arquitectura Modelo-Vista-Template (MVT), buscando así conseguir una correcta organización del código teniendo en cuanta la funcionalidad que proporciona cada elemento de dicha arquitectura.

Por último, el ciclo de vida que se ha seguido para desarrollar el software ha sido el modelo en cascada con retroalimentación, buscando así organizar mejor las fases por las que ha pasado el presente TFG.

# Palabras clave

Aplicación web, Django, Bootstrap, Modelo en cascada con retroalimentación, Arquitectura Modelo-Vista-Template (MVT)

# Abstract

This Project aims to develop a web application to monitor the life cycle of people with autism spectrum disorders, thus replacing the way in which this monitoring is currently carried out; the application is intended for students of the association Alenta.

The Django framework has been used for the web implementation, as well as other tools such as the PostgreSQL database engine and the front-end framework Bootstrap.

The application called **Camino de Vida**, on the one hand, allows the registration of users who are going to follow their life cycle. In this register a user profile is created that contains the information of the students, as well as images of different stages of their development. On the other hand, we have the characteristic component of this application, the monitoring of its life cycle, to carry it out there is a series of tasks, where are stored the assessments that students make regarding their ability to perform the tasks, they must be learning in their life cycle. From these assessed tasks, a life path is generated with a series of goals, where students can visualize their progress in the life cycle.

According to this, the Django project has been modulated in 2 applications: *Accounts* and *Tasks*, which respectively take care of the parts we have just mentioned.

To emphasize that for the design of the application it has followed the architecture Model-View-Template (MVT), looking for this way to obtain a correct organization of the code taking into account the functionality that provides each element of this architecture.

Finally, the life cycle that has been followed to develop the software has been the Iterative Waterfall Mode, thus seeking to better organize the phases through which the present TFG has passed.

# Keywords

Web application, Django, Bootstrap, Iterative Waterfall Model, Architecture Model-View-Template (MVT)

*Agradecimientos*

***En primer lugar, agradecer a mi familia por todo el apoyo que me han dado en los momentos que lo he necesitado desde que empecé el grado.***

***Así como a mi tutor Germán Montoro, por darme la posibilidad de realizar el TFG con él, y por la ayuda que me ha dado siempre que se lo he solicitado. Sin olvidar agradecer el apoyo y facilidades que me han dado las encargadas de Alenta: Guadalupe Montero y Loles Villalva.***

***Por terminar, agradecer a todos mis amigos que he conocido durante el grado, han sido un apoyo fundamental en mi etapa universitaria.***

# INDICE DE CONTENIDOS

# INDICE DE FIGURAS

# INDICE DE TABLAS

# 1 Introducción

En este capítulo se detalla la motivación que ha originado el desarrollo de este Trabajo de Fin de Grado, así como los objetivos que se pretenden conseguir. Además de una breve explicación de cómo se ha organizado el contenido en la presente memoria.

## *1.1 Motivación*

Para entender cómo surgió la idea que motivo la creación de este trabajo, empezaremos hablando de la organización **Alenta**, la cual se dedica a mejorar la calidad de vida y promover el avance de las personas con discapacidad intelectual y otros trastornos del desarrollo, como es el caso de las personas que tienen trastornos del espectro autista. Esta asociación presta diversos servicios a este colectivo, como son: Colegio de Educación Especial, Centro Ocupacional, Centro de Día, Residencia y Vivienda Comunitaria.

Centrándonos en el día a día del Colegio de Educación Especial, los profesores y educadores del centro observaron que había un problema con algunos alumnos que se encontraban en la etapa de transición a la vida adulta, los cuales ya tenían una edad considerable y sin embargo todavía no sabían realizar una serie de tareas de la vida cotidiana que normalmente se aprenden en etapas previas.

Por el motivo anterior, los profesores se pusieron a pensar y desarrollar alguna medida con la cual poder paliar el problema anterior, surgiendo en este proceso la idea de crear un registro individual con las tareas que los alumnos debían ir aprendiendo en las distintas etapas y áreas de su vida, y que fueran los propios alumnos los que valorasen su nivel de habilidad para realizar cada tarea de la lista; además por otro lado se definió una serie de metas que se alcanzan completando ciertos grupos de tareas. Buscando así motivarles al poder ver reflejado su progreso para realizar las tareas.

En el proceso de ir refinando esta herramienta, surgió la idea de desarrollar una aplicación web para integrar esta actividad y satisfacer sus necesidades, y de esta manera poder aprovechar el potencial y el impacto que ofrece la tecnología en el desarrollo de la vida de las personas, concretamente en este caso, realizando una aplicación que ofrezca una serie de características que faciliten el uso de la tecnología por parte de las personas con discapacidad.

Por último, destacar, que a los alumnos les encanta sentir que van mejorando su habilidad para realizar las tareas de la vida cotidiana, así como poder usar una aplicación en lugar de métodos más tradicionales.

## 1.2 Objetivos

Por lo tanto, el objetivo principal de este Trabajo de Fin de Grado es el diseño e implementación de una aplicación web para sustituir y mejorar la manera con el cual se realiza seguimiento del ciclo de vida de personas con trastorno del espectro autista, esto actualmente se realiza en un documento impreso.

En cuanto los objetivos de la aplicación web, podemos distinguir de manera general, los siguientes:

- Ser un sistema de acceso local, ya que la aplicación en principio está únicamente destinada para los alumnos de la asociación **Alenta**, y es por ello que se desplegará en un servidor de manera local para dicha asociación.

  La principal razón de esta decisión es que el sistema va a guardar datos personales e imágenes de los usuarios, y para desplegar una aplicación con esta característica buscando que sea accesible desde cualquier dispositivo a través de Internet se debe garantizar que cumple con la Ley Orgánica de Protección de Datos de Carácter Personal, lo cual tiene una carga de trabajo que puede ser materia de otro proyecto posterior.

- Gestionar la información personal (nombre completo, fecha de nacimiento, género e imágenes personales) de los alumnos.

- Permitir la autoevaluación de tareas y visualización del progreso en su camino de vida.

- Proporcionar ciertas características que faciliten la accesibilidad a las personas para las cuales está destinada.

Mencionar que el nombre que se ha decidido asignar a la aplicación es ***Camino de vida***, y por lo tanto así nos referiremos de ahora en adelante a la aplicación web.

Hay destacar que esta aplicación está adaptada las necesidades especificadas por el cliente (Alenta) durante las distintas reuniones que han tenido lugar.

## 1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 2 (Estado del Arte):** en este capítulo se analiza la situación del Estado del Arte, en primer lugar, se detalla la solución actual que se emplea para realizar el seguimiento del ciclo vida, y posteriormente se analizan una serie de aplicaciones que se aproximan a la desarrollada.

- **Capítulo 3 (Definición del Proyecto):** en este capítulo se detalla la metodología de desarrollo de software que se ha usado para desarrollar esta aplicación, así como una descripción y comparativa de las tecnologías usadas.

- **Capítulo 4 (Análisis):** en este capítulo se detalla la fase de análisis, definiendo el alcance y requisitos de la aplicación.

- **Capítulo 5 (Diseño):** en este capítulo se describe la fase de diseño, en concreto se detalla el diseño de la arquitectura del sistema, la base y la interfaz gráfica.

- **Capítulo 6 (Desarrollo):** en este capítulo se describe las tareas que se han llevado a cabo durante el desarrollo de la aplicación con las herramientas descritas en la fase de definición del proyecto.
- **Capítulo 7 (Integración, pruebas y resultados):** en este capítulo se recoge la batería de pruebas realizadas a la aplicación, así como los resultados de cada una de ellas.
- **Capítulo 8 (Conclusiones y trabajo futuro):** en el capítulo final se reflejan las conclusiones del TFG, además de describir las líneas que en el futuro se pueden seguir para mejorar la aplicación.

# 2 Estado del arte

En este apartado se analizará la solución actual que se emplea para resolver el problema. Además, se incluye un análisis de aplicaciones similares a ***Camino de Vida***, así como una comparativa entre ellas.

## *2.1 Solución actual del problema*

En este punto tuvimos una reunión con los responsables de Alenta, en el cual nos explicaron en detalle la solución que emplean actualmente para llevar el seguimiento del camino de vida de los alumnos, así como las necesidades que cubre, a continuación, pasamos a detallar dicho método.

### 2.1.1 Camino de vida en papel

Actualmente en **Alenta** se emplean dos documentos desarrollados con *Microsoft Word* para realizar el seguimiento del ciclo de vida de cada alumno. Mencionar que la autora de estos documentos es: Mariñe Rodríguez Echevarría.

Por una parte, tendríamos el documento de autoevaluación de tareas las cuales están clasificadas por grupos. Para evaluar dichas tareas se utiliza un código de colores basado en los semáforos, es decir, se utiliza los colores:

- Verde: para evaluar las actividades que el alumno es capaz de realizar sólo y sin haya que recordárselo, o bien simplemente hay que recordárselo una o dos veces.

- Amarillo: para evaluar aquellas tareas que el alumno es capaz de realizar sólo pero hay que recordárselo más de dos veces, o bien porque necesita ayuda física para realizarlas.

- Rojo: para evaluar aquellas tareas que el alumno no es capaz de realizarlas, o en el caso de que no tenga necesidad de realizarlas o físicamente no puede.

En las siguientes imágenes se puede observar el documento en cuestión:


**Figura 2-1: Capturas del documento de autoevaluación (a)**


**Figura 2-2: Capturas del documento de autoevaluación (b)**

Por otra parte, tendríamos el documento de progreso de camino de vida, en el cual se refleja la historia personal de cada alumno con las metas que ha ido alcanzado al completar ciertas tareas, por ejemplo, si es capaz de realizar las tareas T2, T4, T10 habrá completado la meta M1 (estos identificadores son ficticios).

La manera en la que se representa que una meta se ha completado es coloreando el pictograma que identifica la meta en cuestión. Además, los alumnos tienen la posibilidad de añadir fotos personales al inicio y final de cada etapa del camino de vida para personalizarlo. Pueden añadir una foto en las etapas de: recién nacido, foto 6-7 años, foto 12-14 años y foto 20 años.

De igual manera, en las siguientes imágenes siguiente se puede observar este segundo documento:

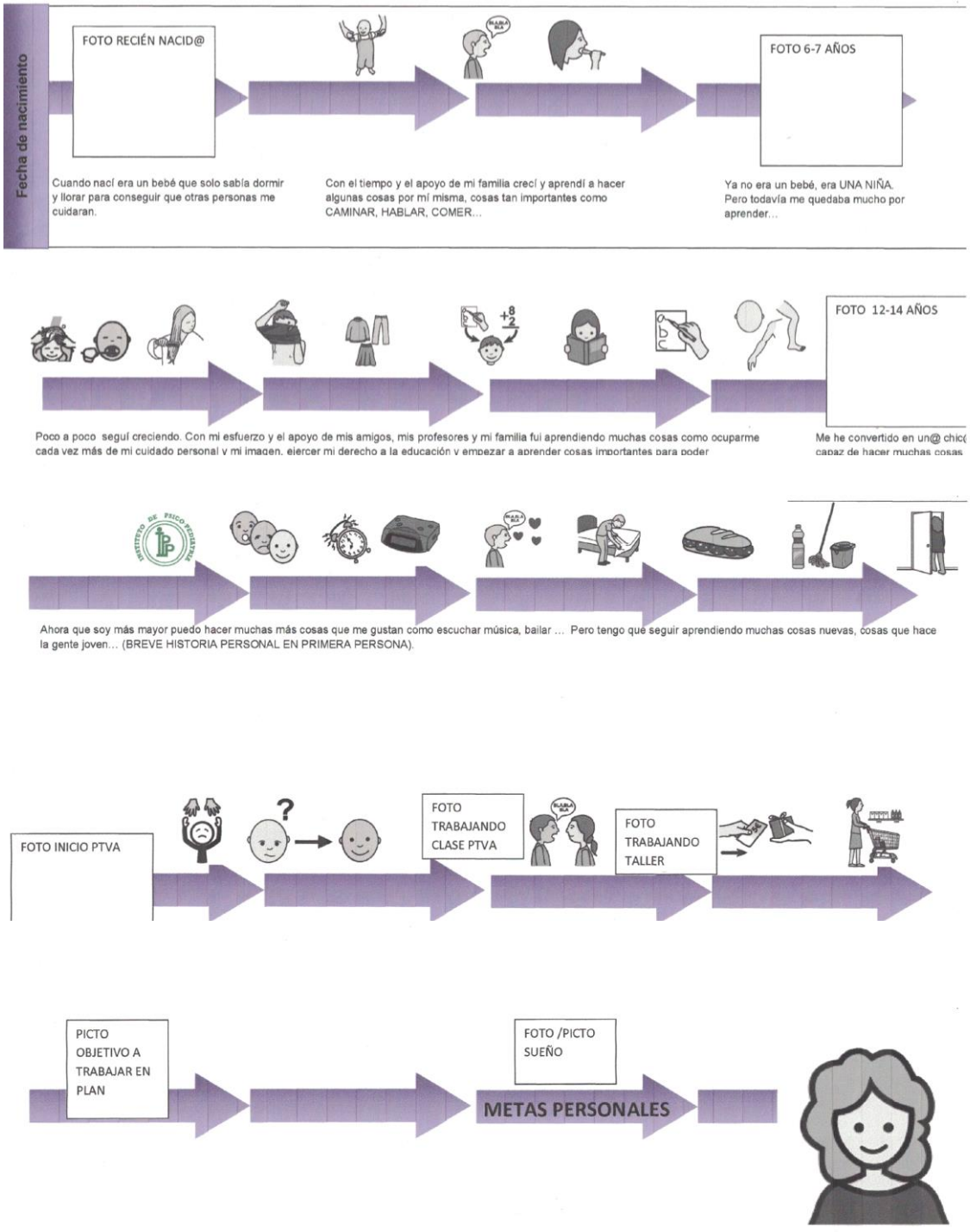


**Figura 2-3. Capturas del documento de progreso del camino de vida**

En cuanto a aplicaciones, vamos analizar tres aplicaciones que son una aproximación en el aspecto de la funcionalidad de definir tareas para alcanzar una determinada meta. Pero como veremos estas no se adaptan completamente a las necesidades que se cubren con la solución actual, y que se pretenden de igual manera satisfacer con la aplicación desarrollada.

### 2.1.2 MyLifePlan

*MyLifePlan* [1] es una aplicación en la cual un administrador puede generar una lista de elementos/tareas a valorar con un grado de satisfacción por parte de cada persona. Con el objetivo de ayudar a las personas a centrarse en los aspectos positivos de su vida, que serán los elementos que el administrador va añadiendo para valorar. Y partir de dichos elementos crear una agenda personal.

Está aplicación ofrece a los usuarios la posibilidad de añadir fotos y comentarios personales a cada elemento que el administrador cree. Otra funcionalidad que ofrece es generar un archivo PDF con las respuestas del usuario.

En las siguientes imágenes se pueden observar capturas de pantalla de la aplicación:

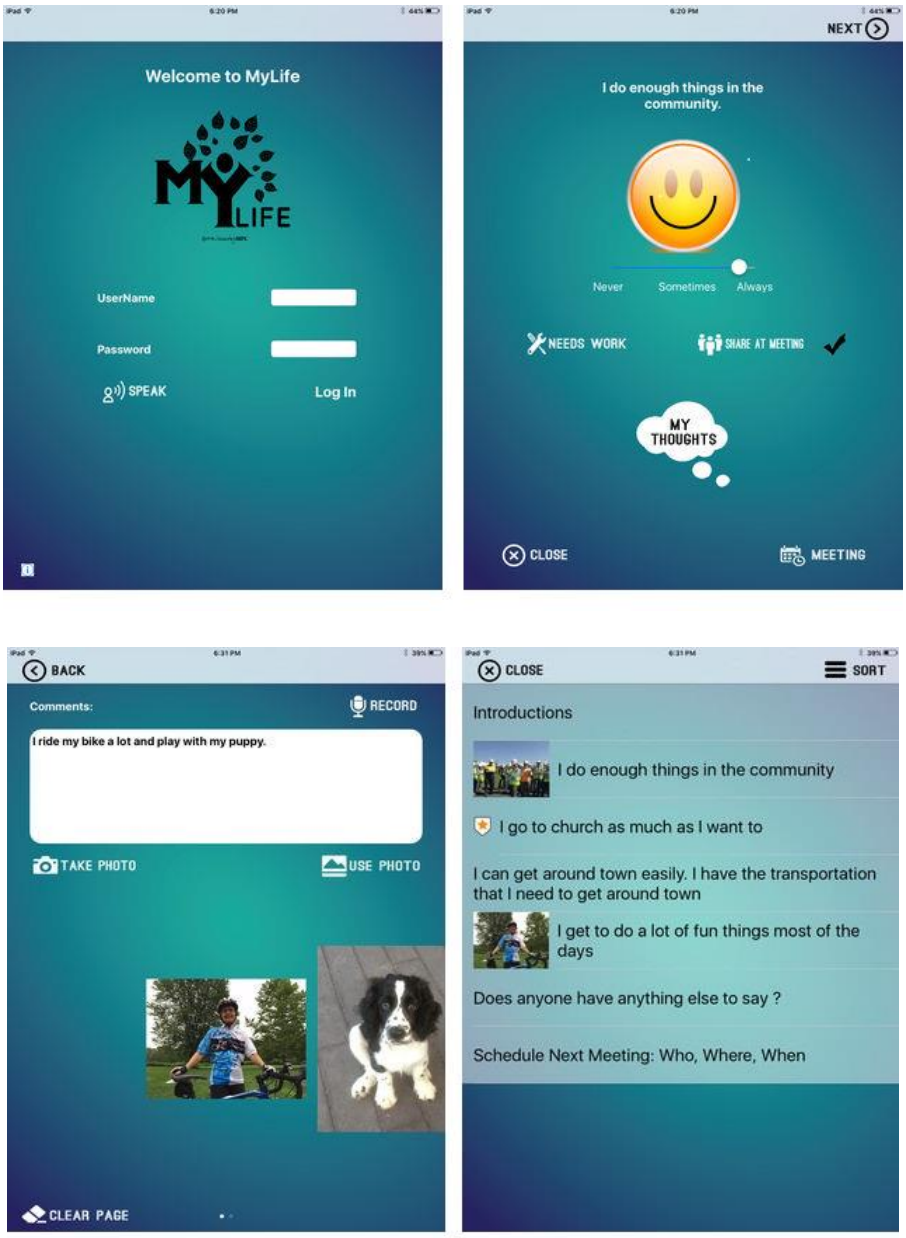


**Figura 2-4: Capturas de la aplicación MyLifePlan**

### 2.1.3 GoalPlanDo

*GoalPlanDo* [2] es una aplicación con el objetivo de que cada usuario planifique un proyecto a largo plazo a partir de un concepto propio. Busca simplificar la gestión de proyectos personales.

Por lo tanto, es el propio usuario el que debe marcar sus objetivos y desarrollar un plan con las tareas que debe ir completando apara alcanzar la meta. Y partir de esa planificación ir marcando las tareas como completadas a medida que las realiza.

La aplicación ofrece la posibilidad añadir comentarios para el plan diseñado, así como poder exportar a PDF o enviar por correo el resumen del plan del proyecto.

En las siguientes imágenes se pueden observar unas capturas de pantalla de la aplicación en funcionamiento:

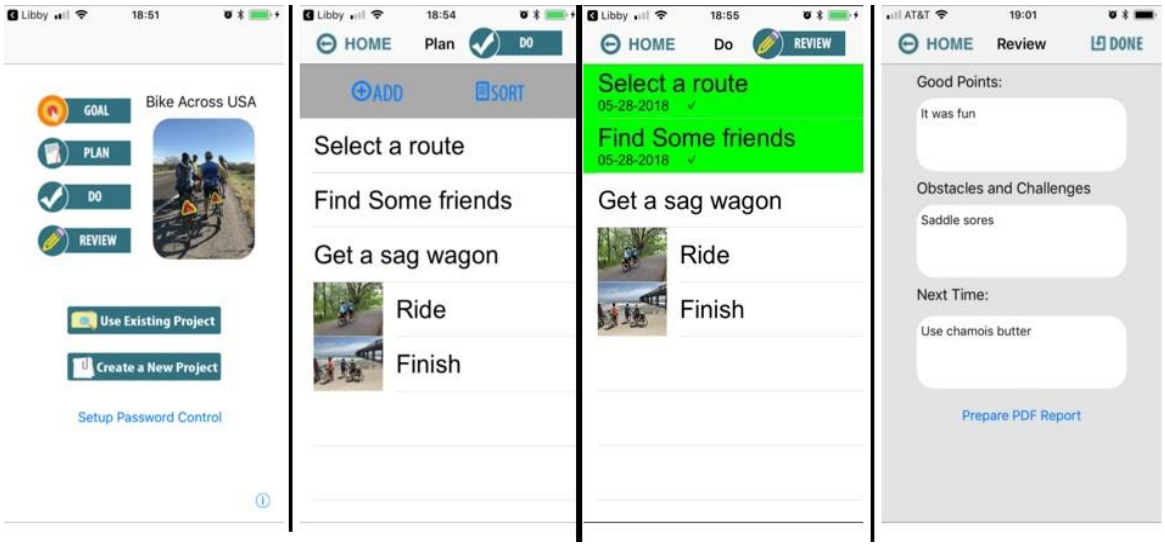


**Figura 2-5: Capturas de la aplicación GoalPlanDo**

### 2.1.4 My Supports by MyKey™ Consulting Services

*My Supports* [3] es una aplicación con el objetivo de ayudar a los usuarios a compartir la información a acerca de sus preferencias personales, intereses, aptitudes y habilidades, con personas de la comunidad que te rodea, puedes decidir con quien compartes o no dicha información.

En la aplicación existen distintas categorías de contenidos con diversas preguntas para responder por parte de los usuarios, entre las categorías que dispone, podemos destacar algunas como: mi comunidad, mi transición y rehabilitación vocacional, mis apoyos en la escuela y mi información personal.

A continuación, podemos observar unas capturas de pantalla de esta aplicación:

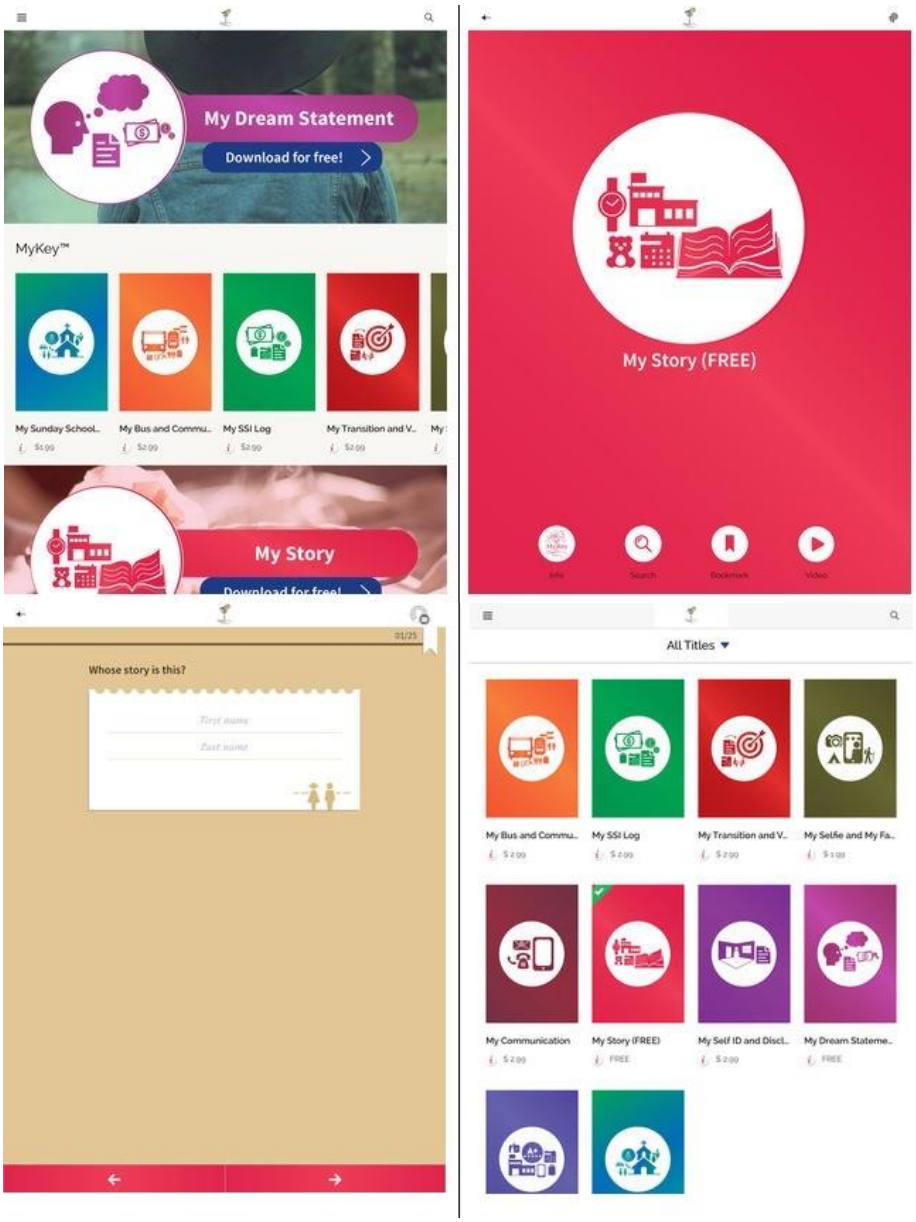


**Figura 2-6: Capturas de la aplicación My Supports**

### 2.1.5 AutiPlan

*AutiPlan* [4] es una aplicación multiplataforma destinada a la planificación diaria de tareas para las personas con autismo, TDAH o PDD-NOS. Busca hacer que su vida diaria sea más predecible a través de la planificación de su rutina diaria, con el objetivo de reducir el estrés y la ansiedad que pueden padecer debido a una vida sin estructurar.

Para ello proporciona la posibilidad de crear horarios visuales para sus tareas y actividades diarias, estas tareas están relacionadas con pictogramas para que su uso sea más sencillo. Tienen la posibilidad de establecer alarmas para recordar las tareas que va a realizar, así como poder generar un documento para imprimir con el horario de tareas y actividades de un determinado día.

En las siguientes capturas podemos observar 2 vistas de la aplicación:

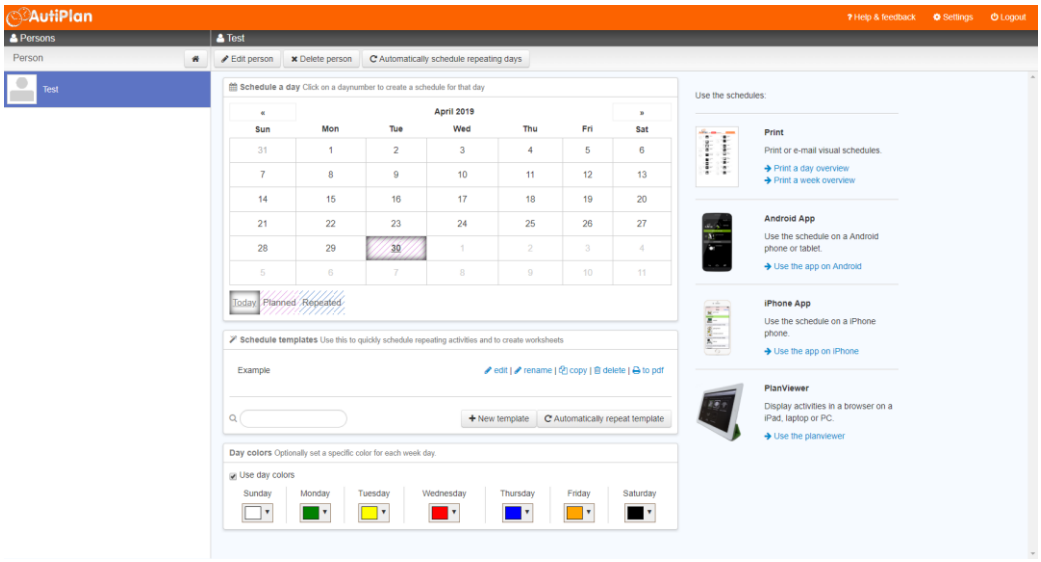


**Figura 2-7: Vista del calendario de la aplicación AutiPlan**

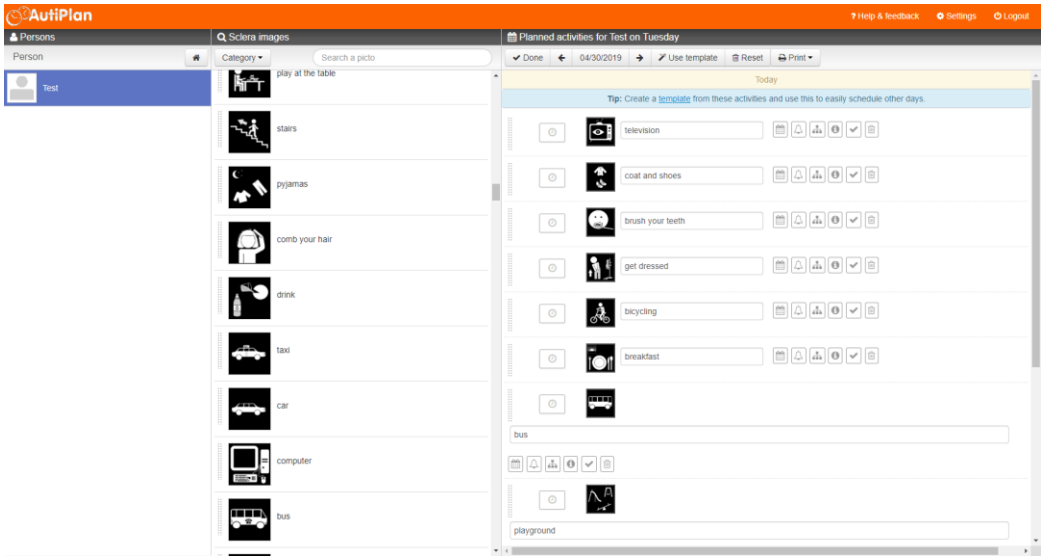


**Figura 2-8: Vista de la herramienta de creación de horarios de la aplicación AutiPlan**

### 2.1.6 Comparativa de las aproximaciones

En este subapartado compararemos distintas características de las aplicaciones que se acaban de analizar, así como las principales diferencias respecto a la aplicación desarrollada en este proyecto.

| | **MyLifePlan** | **GoalPlanDo** | **My Supports** | **AutiPlan** |
|---|---|---|---|---|
| **Dispositivos compatibles** | Dispositivos con sistema operativo iOS | Dispositivos con sistema operativo iOS | Dispositivos con sistema operativo iOS | Navegadores web y Android |
| **Idioma** | Inglés | Inglés | Inglés | Inglés |
| **Diseñado adaptado a personas con discapacidad intelectual** | No | Sí | Sí | Sí |
| **Uso gratis** | Sí | Sí | No (a excepción de categorías de prueba) | No |
| **Exportar progresos a PDF** | Sí | Sí | No | Sí |
| **Contenidos personalizables** | Sí | Sí (a nivel de usuario) | No | Sí (a nivel de usuario) |

**Tabla 2-1: Comparativa de las aproximaciones**

Como conclusión, hemos podido ver que estás aplicaciones cumplen en mayor o menor medida la parte de poder crear un plan de tareas para completar así un objetivo.

No obstante, el contenido que ofrecen no se adapta a la forma de desarrollar el seguimiento del ciclo de vida que se realiza con la solución actual. Las aplicaciones analizadas buscan crear un plan de tareas diario o de corto plazo para ayudar a estructurar la vida de los usuarios, y de alguna manera reducir la incertidumbre de las actividades que van realizar cada día.

En cambio, con **Camino de Vida** se busca marcar una serie de habilidades que los usuarios deben ir alcanzando a lo largo de su desarrollo, y de esta manera servir de referencia para que sepan en que áreas de su vida deben trabajar para alcanzar las metas marcadas.

Por el motivo que acabamos de mencionar, el contenido de la aplicación **Camino de Vida** cambia en gran parte en comparación con las aproximaciones vistas, ya que el contenido que ofrecen persigue otros objetivos, por ejemplo, la aplicación **GoalPlanDo** es el propio usuario el que establece sus metas y lista de tareas, que puede estar relacionado o no con su desarrollo a lo largo de las etapas de su vida.

En contraste, el contenido que incluye **Camino de Vida** es el que ha sido desarrollado por los profesores de Alenta, el cual busca servir de referencia en cuanto a las habilidades que los alumnos deben ir adquiriendo a lo largo de sus distintas etapas de desarrollo. Y de esta manera si algún usuario se queda rezagado en alguna habilidad, los profesores y su familia puedan centrarse en ayudarles a progresar en esas tareas específicas.

Por otra parte, una gran desventaja de las aplicaciones analizadas es el idioma en el que están desarrolladas, todas tiene como idioma principal el inglés, lo que las convierte en una opción inviable a usar con los alumnos de la asociación Alenta, ya que este grupo de usuarios tiene como lengua principal el español. Por lo tanto, el contenido de la aplicación **Camino de Vida** estará en español.

Otra característica que diferencia a la aplicación **Camino de Vida** de las aplicaciones analizadas, es que dispondrá de una funcionalidad que permite ver de manera sencilla y visual el progreso tiene cada usuario en sus habilidades a lo largo de sus etapas de desarrollo. En las aplicaciones analizadas el seguimiento de las tareas solo se centra en una meta específica, no hay ningún tipo de relación entre tareas de 2 metas distintas. Una vez completado un objetivo establecido (por el propio usuario o por un administrador), esa meta simplemente se archiva, no se realiza un seguimiento del usuario a lo largo de su vida.

Para terminar, compararemos las características de accesibilidad que incorpora cada aplicación, obviando a **MyLifePlan** ya que su diseño no está específicamente desarrollado para personas con discapacidad intelectual. En cuanto a las otras alternativas, todas incorporan características para mejorar su uso, como son buen contraste en los colores usados, tamaño de la fuente y de los pictogramas lo suficientemente grande. La aplicación **Camino de Vida** de igual manera incorpora esas características, y además se le ha añadido la posibilidad de poder elegir que los textos se muestren en mayúsculas, así como la posibilidad de elegir el tipo de fuente usada, existen 2 posibilidades: la fuente ‘arial’ (por defecto) y la fuente ‘opendyslexic’ que un tipo de fuente diseñada para mejorar la lectura de las personas con dislexia.

# 3 Definición del proyecto

En este apartado se define la metodología de desarrollo de software que se ha usado a lo largo del desarrollo de este proyecto. Además de incluir un listado con las herramientas que se han usado para desarrollar la aplicación.

## *3.1 Metodología*

El ciclo de vida elegido para desarrollar este trabajo ha sido el **Ciclo de vida en cascada con retroalimentación**, el cual nos determina una serie de etapas para desarrollar el software, permitiendo separar todas las actividades involucradas a lo largo del desarrollo de la aplicación.

Esta metodología consta de las siguientes etapas:

1. **Análisis**: Es la fase inicial del proyecto, durante la cual se determinan los requisitos y objetivos que debe alcanzar la aplicación. En esta etapa se colabora directamente con el cliente para captar sus necesidades respecto a la aplicación. Es por tanto una etapa muy importante, ya que es esencial definir con precisión los requisitos.

2. **Diseño:** En esta fase se realizan todas aquellas actividades relacionadas con el diseño de la aplicación en todos sus niveles. Esto incluye el diseño arquitectónico, el diseño de la base de datos, el diseño de las pantallas y otros menos notables como puede ser el diseño de la estructura de navegación dentro de la aplicación.

3. **Implementación:** En esta fase se desarrolla todo el código ejecutable y configuraciones necesarias para implementar la aplicación de acuerdo al trabajo realizado en las 2 primeras fases. Esta fase es la que más tiempo requiere hasta que se despliega la aplicación.

4. **Pruebas e integración:** En esta fase se realizan una batería de pruebas para poder garantizar en cierto grado que la etapa de implementación se ha completado con éxito. Como resultado de esta fase se ha documentado las pruebas realizadas, así como su resultado.

   Una vez se ha superado la parte de las pruebas, se procede al despliegue de la aplicación en el entorno de producción, que en este caso ha sido un servidor local en Alenta.

5. **Mantenimiento:** Es la fase final del ciclo de vida del software, a lo largo de esta etapa se mejora y actualiza todo aquello que sea necesario. Esta etapa se puede extender mucho en el tiempo, todo depende de las exigencias y uso final que haga el cliente. Esta etapa quedará como trabajo futuro.

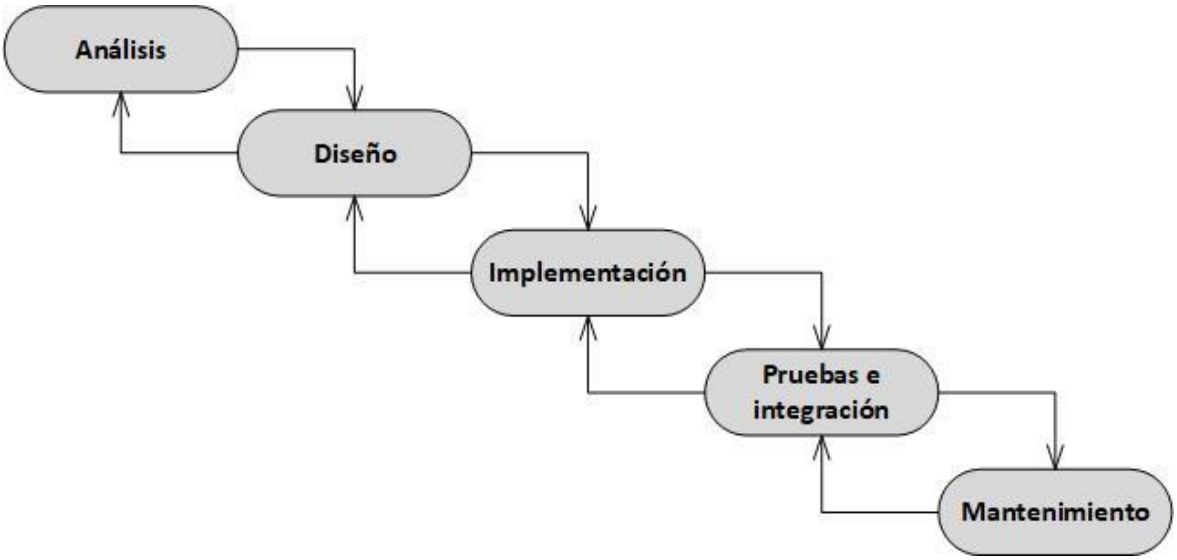


**Figura 3-1: Ciclo de vida en cascada con retroalimentación**

El motivo por el cual se ha elegido esta metodología es porque nos permite volver a etapas anterior las veces que sea necesario, para poder así corregir errores que se hayan producido en etapas previas.

Por ejemplo, si en la fase de implementación nos percatamos que hemos cometido un fallo en el diseño de la base de datos, perfectamente podríamos volver a iterar sobre la fase de diseño para corregir lo que necesitemos. Evitando así limitaciones de otros modelos de desarrollo más clásicos, como puede ser el ciclo de vida lineal, en el cual cada etapa se realiza una sola vez, ya que está pensado para proyectos que están claramente definidos y es poco probable que cambien sus requisitos en el futuro.

Para ayudarnos a la hora de decir el modelo a utilizar, hemos usado las diapositivas de Proyectos de Desarrollo Software [5].

## *3.2 Herramientas utilizadas*

En esta sección se incluye el análisis de las tecnologías que se han empleado para el desarrollo de la aplicación web. Además, se incluyen comparativas con las alternativas que se han tenido en cuenta en cada caso, así como los motivos por los cuales se ha elegido cada tecnología.

### 3.2.1 Django

Django [6] es un *framework* de código abierto para aplicaciones web que está completamente desarrollado en Python. Sigue la filosofía de *batteries-included*, lo que significa que incluye funcionalidades y características comunes en la construcción de aplicaciones web, por ejemplo, incluye una librería para la autenticación de usuarios, enrutamiento de URL's, *testing* de la aplicación, un sitio administrativo y muchas otras tareas comunes de las aplicaciones web.

Como alternativa a Django se estudió el *frameworks* Laravel [7], comparando las siguientes características:

| |  |  |
|---|---|---|
| **Lenguaje de programación** | Python | PHP |
| **Seguridad** | Alta<br><br>Proporciona protección contra inyección SQL, CSRF, clickjacking, etc. | Baja<br><br>Solo proporciona características básicas para la protección de los usuarios |
| **Rendimiento** [8] | 70000 peticiones JSON/segundo | 8500 peticiones JSON/segundo |
| **Patrón de diseño** | MVT/MVC | MVC |

**Tabla 3-1: Comparativa Django y Laravel**

Las razones por las cuales nos hemos decantado por Django son:

- El lenguaje de programación es Python, el cuál es sencillo de aprender, además tiene una sintaxis clara, es muy flexible y fácil de entender. Esto permite que el desarrollo de la aplicación web sea más rápido.
- Utiliza la filosofía de *batteries-included*, que como hemos mencionado anteriormente incluye funcionalidades y características comunes en la mayoría de las aplicaciones web. Esto también facilita el desarrollo, ya que invirtiendo relativamente poco tiempo podemos integrar aquellas funcionalidades comunes que queramos, y así poder centrarnos en las funcionalidades que caracterizan nuestra aplicación.
- Las aplicaciones desarrolladas con Django son muy escalables, ya que usa la arquitectura *shared-nothing*, en la cual se puede agregar *hardware* a cualquier nivel (servidores de bases de datos, servidores de almacenamiento o servidores de aplicaciones) al estar en capas claramente separadas.
- Tiene un nivel de seguridad alto. El *framework* proporciona protección contra inyección SQL, clickjacking, ataques CSRF, etc.

### 3.2.2 PostgreSQL

PostgreSQL [9] es un sistema gestor de bases de datos relacionales que está orientado a objetos, su desarrollo lo lleva la comunidad PGDG (PostgresSQL Global Development Group), la cual lo hace de manera desinteresada apoyada con contribuciones de grupos comerciales.

Django nos proporciona 4 motores de bases de datos integrados con el *framework*: PostgreSQL, MySQL [10], SQLite3 [11] y Oracle [12]. Estás 4 alternativas son la que hemos analizado en la comparativa:

| | PostgreSQL | MySQL | ORACLE DATABASE | SQLite |
|---|---|---|---|---|
| **Modelo primario del sistema** | Relacional | Relacional | Relacional | Relacional |
| **ACID** | Sí | Sí | Sí | Sí |
| **Multisistema** | Sí | Sí | Sí | Sí |
| **Escalabilidad** | Alta | Alta | Alta | Baja |
| **Uso gratuito** | Sí | Sí | No | Sí |

**Tabla 3-2: Comparativa PostgreSQL, MySQL, Oracle Database y SQLite**

Los motivos por los cuales nos hemos decantado por usar el gestor de base de datos PostgreSQL son:

- Cumple completamente las características de Atomicidad, Consistencia, Aislamiento y Durabilidad (siglas ACID en inglés). Por lo cual nos garantiza que los datos almacenados son sólidos gracias a que las transacciones cumplen estas características.
- Cumple con el estándar SQL en gran medida, por lo cual tenemos la posibilidad de usar consultas y reutilizar scripts de otros sistemas de bases de datos que también cumplan con el estándar.
- Tiene una excelente escalabilidad vertical, se nos permite configurar el gestor de acuerdo con el hardware disponible en el equipo. Por los cual se puede ajustar para lograr el número óptimo de transacciones simultaneas.
- Tiene un gran soporte, hay una gran comunidad detrás de este gestor de base de datos. De hecho, se considera que tiene mejor soporte que otros gestores propietarios.

Por último, hay que destacar que Django incluye un API que nos abstrae del nivel de la base de datos, ya que las operaciones de creación, recuperación, actualización y eliminación de objetos se realizan a través de la API.

### 3.2.3 Bootstrap

Bootstrap [13] es un *framework* orientado al desarrollo *front-end*, es decir, orientado a mejorar el diseño e implementación de la parte del cliente, en concreto de las pantallas que usa el usuario. El diseño de este *framework* fue empezado por Twitter, aunque posteriormente fue liberado para licencia MIT para pasar a ser una herramienta de desarrollo web de código abierto.

Este *framework* incorpora múltiples características que facilitan el desarrollo de aplicaciones web, como son los componentes (listas, tablas, desplegables, etc.) los cuales los podemos usar directamente en las páginas, facilitando así la codificación de las pantallas diseñadas.

Además, incluye una serie de utilidades que nos facilitan otros aspectos de la página web, con son el uso de clases css para personalizar los componentes utilizados, como puede ser modificar su tamaño o color. También nos incluye otras utilidades que nos facilitan organizar el *layout* de las páginas, o modificar el espaciado, tamaño, visibilidad, etc. de los distintos elementos de la página.

Como alternativa a Bootstrap se analizó **Semantic UI** [14], que de igual manera es un *framework* orientado al diseño e implementación de las pantallas. El cual tiene como principales ventajas: el uso de un lenguaje natural en el código, una amplísima gama de componentes y una interfaz moderna. Tras compararlo, vemos que las ventajas en comparación con Bootstrap no son significativas.

Por lo tanto, finalmente nos hemos decantado por Bootstrap, por los siguientes motivos:

- La comunidad que hay detrás en su desarrollo es mucho más amplia que la de Semantic UI.
- Es un proyecto mucho más maduro, ya que cuenta con más años de desarrollo.
- Proporciona una cantidad de componentes más que suficiente para desarrollar la aplicación.
- Incorporar múltiples clases CSS que podemos utilizar en nuestros componentes para personalizarlos.

# 4 Análisis

En este capítulo se presenta el resultado de la fase de análisis del proyecto. Para ello se exponen los roles de usuario existentes en la aplicación, así como diagramas con los casos de usos relacionados a cada tipo de usuario.

También se incluye un listado con los requisitos que debe cumplir la aplicación para considerar que se ha completado con éxito. Además de esto, el listado nos sirve como guía para diseñar la batería de pruebas que se realizarán posteriormente. Este el motivo por el cual esta fase del ciclo de vida de la aplicación es de las más relevantes.

Cabe destacar que para el desarrollo de esta fase ha sido fundamental una serie de reuniones con los responsables de Alenta, en el cual nos explicaron las necesidades que debía cubrir la aplicación, ya que ellos son los clientes que definen las funcionalidades de la aplicación, sentando así las bases del proyecto. Este proceso fue interactivo en el cual fuimos mostrando los requisitos capturados en dichas reuniones para comprobar si se adaptaban a lo requerido.

## *4.1 Roles de usuario*

Los usuarios del sistema se pueden clasificar en los dos siguientes grupos:

- **Alumnos**: son los usuarios principales del sistema, ya que son los que hacen usan la aplicación para registrar sus progresos en el camino de vida.

- **Administrador**: es un tipo de usuario que se podría denominar auxiliar, ya que su principal función va a ser prestar soporte a los alumnos en caso de que estos olviden sus credenciales de acceso.

## *4.2 Casos de uso*

En los siguientes diagramas de casos de uso [15] se pueden ver representado las funcionalidades del sistema que tendrá a su disposición cada tipo de usuario:

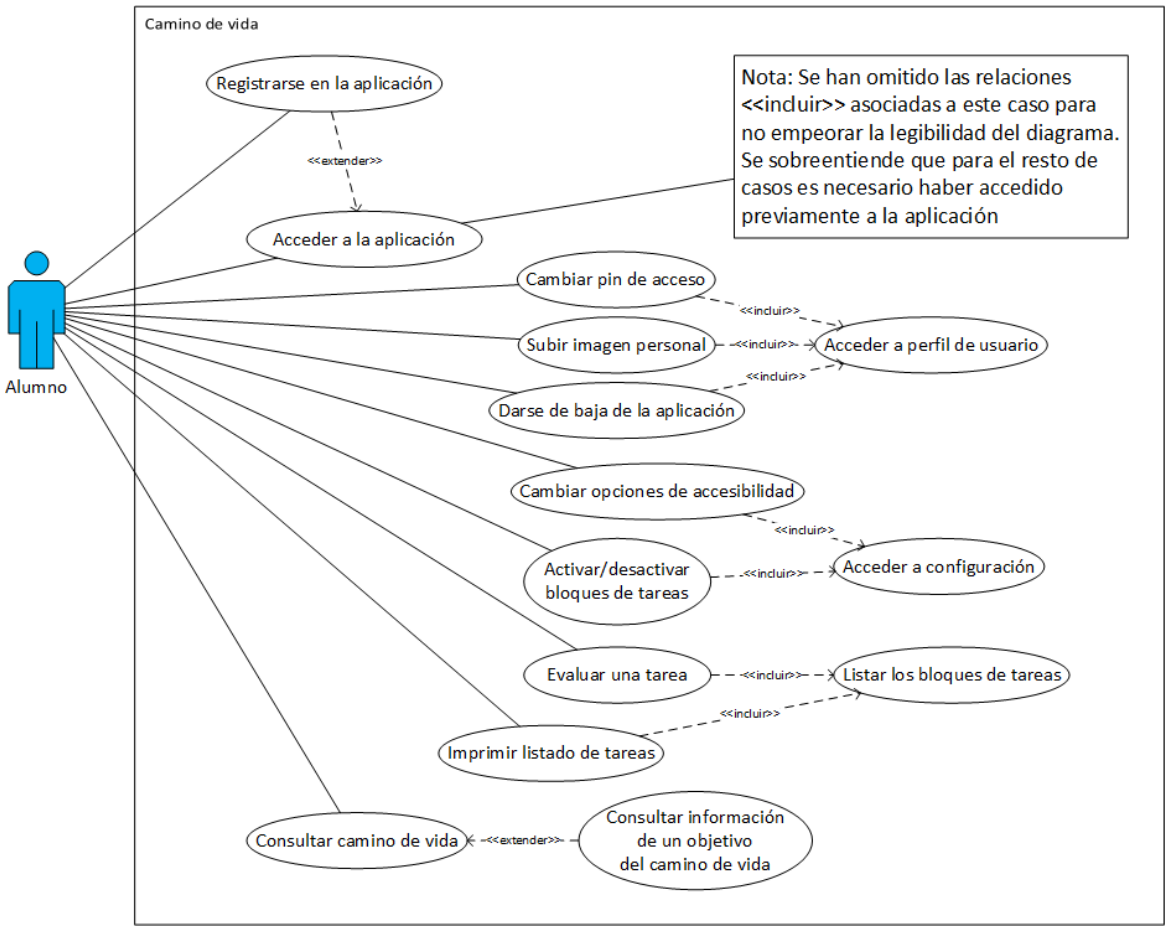


**Figura 4-1: Diagrama de casos de uso para el rol de Alumno**

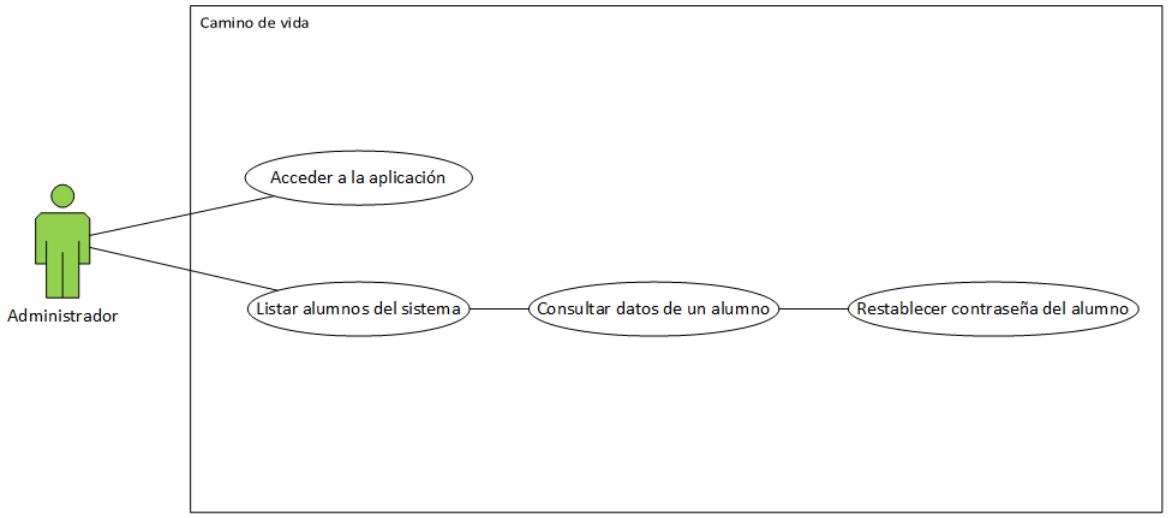


**Figura 4-2: Diagrama de casos de uso para el rol de Administrador**

## *4.3 Listado y definición de requisitos*

Los requisitos de la aplicación se han clasificado en los siguientes dos grupos:

**Requisitos funcionales:** son aquellos que describen las funcionalidades que proveerá el sistema.

**Requisitos no funcionales:** son aquellos que no especifican directamente las funcionalidades propias del sistema, sino que describen propiedades de la aplicación. Atendiendo aspectos como por ejemplo la seguridad, el rendimiento, la usabilidad, disponibilidad, etc.

### 4.3.1 Requisitos funcionales

**RF-1.- Registro**
Para utilizar la aplicación se solicita a los usuarios que se registren, para ello será necesario que introduzcan los siguientes datos personales: nombre, fecha de nacimiento, usuario, pin.

**RF-2.- Autenticación**

> **RF-2.1.- Alumnos**
> Para poder usar las funcionalidades asociadas a este rol de usuario será necesario acceder a la aplicación con las credenciales que se indicaron a la hora de registrarse en la página correspondiente.
>
> **RF-2.2.- Administrador**
> De igual manera, el administrador, deberá acceder a la aplicación con las credenciales proporcionadas por el desarrollador de la aplicación para poder hacer uso de sus funciones. Mencionar que una vez haya accedido podrá cambiar su contraseña de acceso.

**RF-3.- Consultar perfil de usuario**
Los alumnos tendrán una sección donde podrán consultar su perfil, allí podrán ver los datos personales que la aplicación almacena para identificarlos.

**RF-4.- Cambiar pin de acceso**
Los alumnos deben tener la posibilidad de poder cambiar su pin personal de acceso.

**RF-5.- Subir imágenes personales**
La aplicación debe permitir que los alumnos puedan subir imágenes personales, que se corresponderán a las distintas etapas de su camino de vida.

**RF-6.- Configurar los bloques de tarea**
Cada alumno tendrá la posibilidad de definir los bloques de tarea que la aplicación le mostrará, pudiendo así adaptar las tareas visualizadas al trabajo que están realizando en cada momento.

**RF-7.- Listar los bloques de tareas**
Los alumnos podrán listar los bloques de tareas disponibles según su configuración, para poder consultar las tareas que deben ir completando o bien para hacer la autoevaluación de alguna tarea en concreto.

**RF-8.- Autoevaluar una tarea**
La aplicación debe permitir a los alumnos autoevaluarse en las tareas, para ello se usará un código de colores basado en los semáforos: rojo (no es capaz de hacer una tarea), amarillo (necesita ayuda para hacer la tarea) y verde (puede realizar la tarea por sí mismo).

**RF-9.- Consultar camino de vida**
Los alumnos tendrán una sección para consultar sus progresos del camino de vida de una manera gráfica. Esta sección mostrara las tareas clasificadas por etapas, además de las fotos de cada etapa que el alumno haya subido a su perfil.

**RF-10.- Obtener información de como completar una tarea**
En la sección donde se muestra el progreso del camino de vida, se debe ofrecer la posibilidad de obtener información de que tareas son necesarias completar para alcanzar una determinada meta del camino de vida.

**RF-11.- Imprimir listado de tareas con su evaluación**
Los alumnos tendrán la posibilidad de imprimir el listado de tareas con su evaluación correspondiente, generando un documento similar al actual donde ellos se autoevalúan.

**RF-12.- Dar de baja a un usuario**
El administrador podrá dar de baja de manera temporal o definitiva a un usuario de la aplicación.

### 4.3.2 Requisitos no funcionales

**RNF-1.- Opciones de accesibilidad**
La aplicación debe ofrecer la posibilidad de cambiar algunas opciones de accesibilidad, tales como el tipo y tamaño de la fuente.

**RNF-2.- Compatibilidad con distintas plataformas**
La aplicación se debe adaptar correctamente tanto a *tablets* como a ordenadores, ya que son los dispositivos a través de los cuales los usuarios accederán a la aplicación web.

**RNF-3.- Conexión a la red local**
Tanto los alumnos como el administrador usaran la aplicación dentro de una red local, está red será la del centro donde se despliega la aplicación.

**RNF-4.- Seguridad**
La aplicación debe garantizar que los datos personales de los usuarios se almacenan de una forma segura.

# 5 Diseño

En esta sección se detalla la fase diseño de la aplicación *Camino de vida*. En primer lugar, se incluye la descripción de la arquitectura general del sistema, luego pasaremos a detallar el patrón de diseño que se ha usado para desarrollar la aplicación, denominado Modelo Vista Controlador (MVC), aunque como veremos luego Django hace una interpretación propia de dicho patrón.

También se aborda la manera en la que se estructuran los datos que se almacenan en la base de datos PostgreSQL.

Y para terminar esta sección se muestra el diseño de la interfaz gráfica de la aplicación, así como un diagrama de flujo entre las distintas páginas de la aplicación web.

En esta etapa del ciclo de vida también fue necesario hacer algunas reuniones con los responsables de Alenta, en donde les enseñamos principalmente las maquetas del diseño gráfico realizado, para así poder recibir sugerencias en cuanto a cambios que se debían efectuar en las pantallas de la aplicación, para que estas se adapten a las necesidades del tipo de usuarios a los que va dirigida.

## *5.1 Arquitectura general de la aplicación*

Basándonos en los requisitos del sistema, se utilizará una arquitectura de tipo cliente-servidor, la cual es la arquitectura más común en lo que se refiere a aplicaciones web. Dicho modelo básicamente es un sistema de información distribuido, en el cual uno o más clientes hacen uso de servicios y recursos proporcionado por el servidor.

Por lo tanto, en este modelo de diseño de software podemos diferencias los siguientes elementos:

- **Clientes**: son los usuarios que usan los servicios y recursos ofrecidos por el servidor, en el caso de nuestra aplicación lo harán a través de un navegador web desde un ordenador o una *tablet*.

- **Red**: es el medio de comunicación entre los clientes y el servidor, en nuestro caso se hará a través de una red local del centro Alenta, ya que la aplicación se usará por los alumnos de este centro.

- **Servidor**: son los dispositivos/procesos que brindan los servicios y recursos a los clientes. El flujo habitual es, que el cliente realiza una petición en el navegador web, se transmite a través de la red, el servidor atiende los requerimientos del cliente y le devuelve la respuesta a través de la red.

A continuación, podemos ver un diagrama con la arquitectura general de nuestro sistema web:

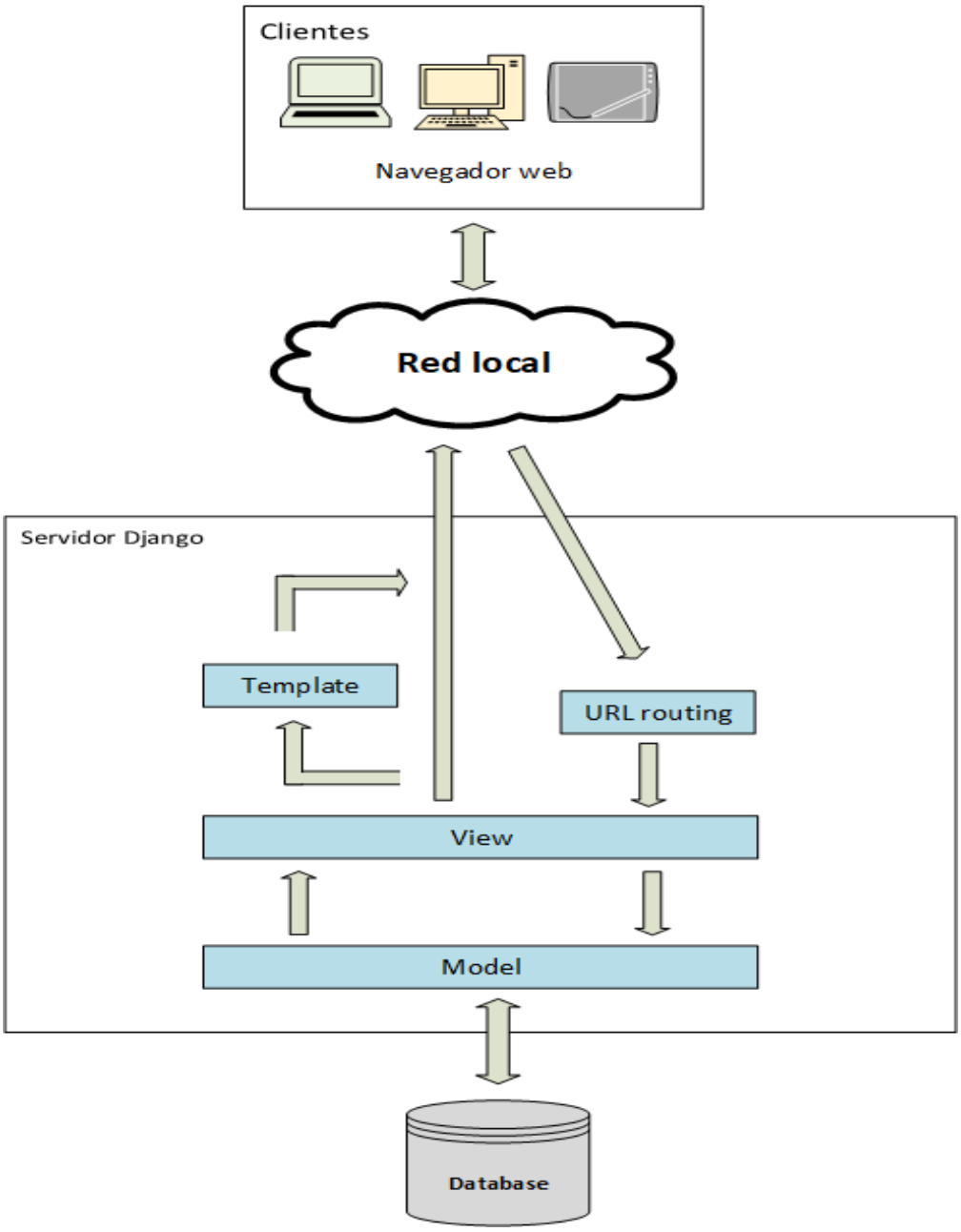


**Figura 5-1: Diagrama de la arquitectura del sistema web.**

## *5.2 Arquitectura del servidor*

Para la arquitectura del servidor se ha utilizado el patrón de diseño MVC [16], como se ha mencionado anteriormente *Django* hace una interpretación propia de dicho patrón, pasando a denominarse *Model Views Templates* (MVT) [17].

Antes de explicar en detalle la función de cada componente, vamos a ver la analogía que hay entre el patrón MVT y el patrón original MVC:

- El componente *modelo* mantiene la denominación.
- El componente *controlador* pasa a denominarse *vista*.
- El componente *vista* pasa a denominarse *plantilla* (*template).*

El **modelo** es el elemento de acceso a la base de datos, esta capa contiene toda la información de cómo se organizan las entidades/tablas que componen la base de datos, así como las relaciones entre ellas. Además, contiene las validaciones que debe superar la información a almacenar en cada campo de las tablas.

La **vista** es el elemento que contiene la lógica de negocios, es decir, es la capa que accede al modelo para obtener la información correspondiente a cada petición, para posteriormente delegar al *template*.

El ***template*** es la capa que se encarga de la presentación de la información proporcionada por la vista.

### 5.2.1 Base de datos

Para el diseño de la base de datos usaremos un diagrama entidad-relación, en dicho diagrama se representan las entidades y relaciones que compondrán la base de datos de la aplicación.

En nuestro caso nuestro modelo está formado por:

- **Usuarios:** esta entidad modela la información básica de un usuario de la aplicación, en nuestro caso usaremos el modelo de autenticación de Django.
- **Perfil de usuario:** esta entidad modela la información adicional de un usuario, como es el nombre completo, la fecha de nacimiento, el género y las imágenes personales. En la implementación usaremos una relación *OneToOneField*¸ para usar la entidad Usuario como clave foránea y que la relación entre ambas entidades sea uno a uno.
- **Grupo:** esta entidad modela la información de los grupos de contenidos que formarán la aplicación. Almacenará el título, categoría y estado (habilitado o deshabilitado) de cada grupo. En este caso se usará la entidad Usuario como clave foránea para relacionar a que usuario de la aplicación pertenece cada grupo.
- **Tareas:** esta entidad modela la información de las tareas que componen a cada grupo de contenidos. En concreto en está tabla de la base de datos se almacena el título, evaluación y nombre del pictograma correspondiente a cada tarea. Esta tabla usa como clave foránea la entidad Grupo, para determinar a qué grupo pertenece cada tarea y a través del grupo poder determinar a qué usuario pertenece.

A continuación, se incluye el diagrama entidad-relación [18]:

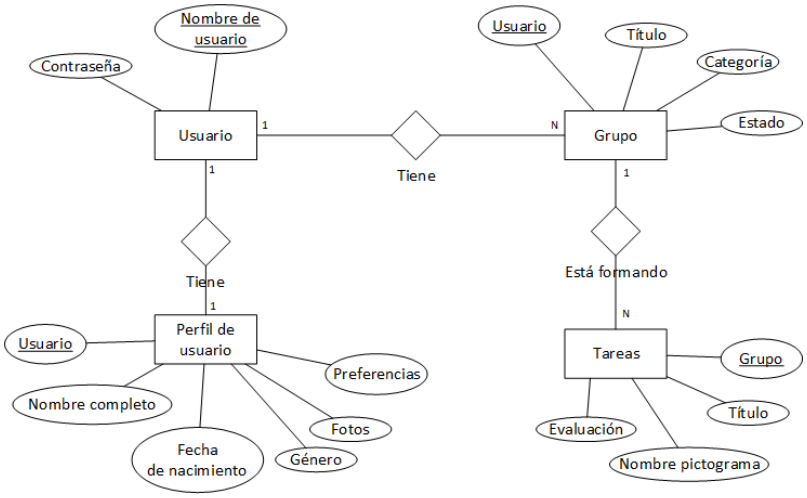


**Figura 5-1: Diagrama entidad-relación**

## *5.3 Interfaz gráfica*

Para el diseño de las maquetas de las pantallas de la aplicación se ha utilizado la aplicación Balsamiq Mockups 3 [19]. En el diseño se ha buscado que las pantallas sigan una estructura común con el objetivo de facilitar la navegación a los usuarios. Las maquetas realizadas se incluyen en el **Anexo A**.

Para complementar este apartado, se incluye el siguiente diagrama de navegación en el cual se puede ver cómo interaccionaría un usuario con las pantallas que forman la aplicación:

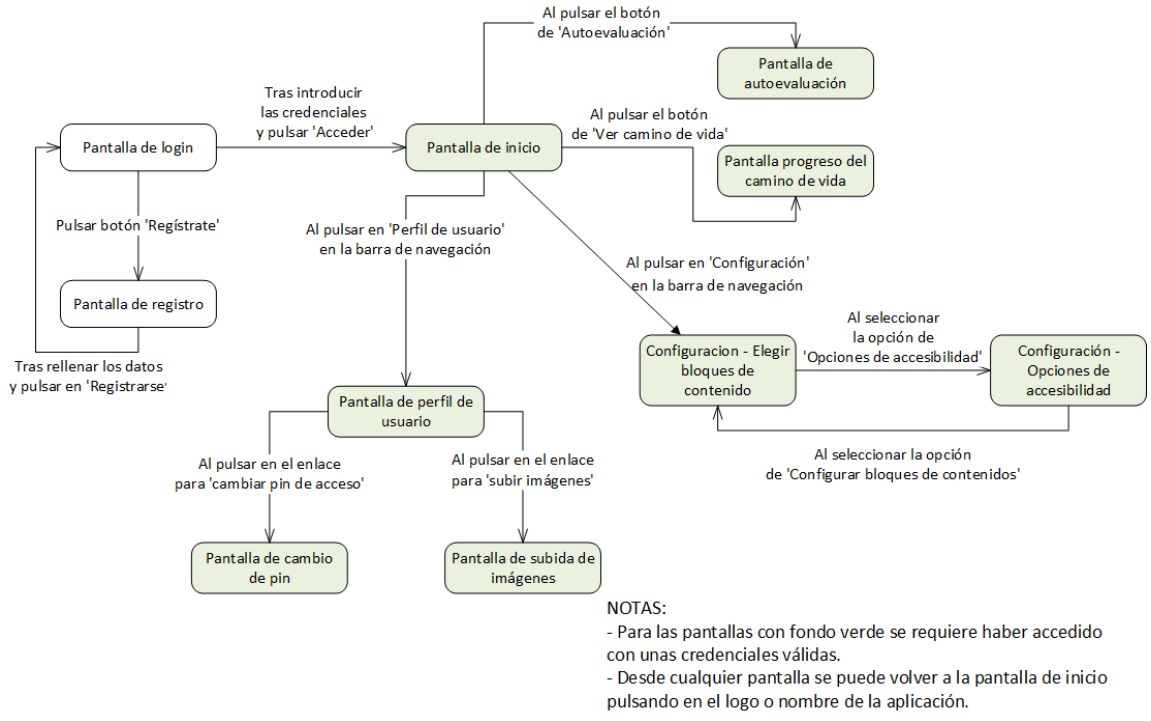


**Figura 5-2: Diagrama de navegación de la aplicación**

# 6 Desarrollo

En este apartado se detallarán las tareas realizadas durante toda la fase de desarrollo de la aplicación. Además, se describirá la estructura en la cual se ha organizado el código desarrollado.

## *6.1 Configuración del entorno de programación*

El primer paso para proceder con la programación de la aplicación consistió en configurar el entorno de desarrollo.

Como base del entorno de desarrollo se decidió utilizar el sistema operativo Ubuntu 18.04 y el editor de código fuente Visual Studio Code. El motivo por el cual me decante por estás opciones es porque son las herramientas con las cuales me siento más cómodo para desarrollar software. Mencionar que el sistema operativo Ubuntu incluye Python instalado de serie, dicho lenguaje de programación es el usado por el *framework* Django.

Una vez tomada está decisión, procedimos a descargar y configurar los paquetes de Python necesarios, para ello usamos la herramienta *Python Package Index* (PyPI) [20], que es el repositorio oficial de aplicaciones y paquetes desarrollados en Python.

El primer paquete que descargamos y configuramos fue **virtualenv**, el cual nos permite crear entornos virtuales de desarrollo con Python. Obteniendo así un gran beneficio, ya que de esta manera podemos crear un entorno virtual para cada aplicación que estemos desarrollando, evitando así conflictos entre las dependencias de paquetes de cada aplicación.

Una vez creamos el entorno virtual, se procedió a instalar dentro de este, los paquetes necesarios para el desarrollo de la aplicación:

- **django 2.1.4:** Es el paquete que incorpora el *framework* con el cual vamos a desarrollar la aplicación.

- **psycopg2 2.7.6.1:** Es el paquete que incorpora los módulos necesarios para que Django pueda trabajar con una base de datos PostgreSQL.

- **pillow 6.0.0:** Es una librería para trabajar con imágenes. En nuestro caso será requerida para poder crear formularios con la posibilidad de subir imágenes por parte del usuario.

Para terminar, instalamos el sistema gestor de base de datos **PostgreSQL** en su versión 10.6, que es el sistema que almacenará la información de la aplicación, como ya hemos comentado en apartados anteriores.

## *6.2 Programación de la aplicación*

El desarrollo de la aplicación comenzó con la creación del proyecto de Django, el cual tiene la siguiente estructura:

```
caminodevida/
    manage.py
    CaminoDeVida/
          __init__.py
          settings.py
          urls.py
          wsgi.py
    accounts/…
    tasks/…
    media/…
```

En primer lugar, tenemos el directorio raíz **caminodevida/** que hace de contenedor de todos los archivos del proyecto.

Dentro del contenedor tenemos los siguientes archivos y directorios:

- El archivo **manage.py** que contiene los *scripts* para poder trabajar con el proyecto de Django, nos permite interactuar a través de la línea de comando. Por ejemplo, para realizar migraciones de los modelos a la base de datos o bien para arrancar el servidor.

- El directorio **CaminoDeVida/** que es el paquete de Python creado para nuestro proyecto.

- El archivo **CaminoDeVida/__init__.py**, el cual es necesario para que Python considere el directorio anterior es un paquete.

- El archivo **CaminoDeVida/settings.py** donde se define la configuración del proyecto.

- El archivo **CaminoDeVida/urls.py,** en el cual se definen las direcciones URL que formarán el proyecto. Posteriormente a este fichero se importarán las URL de las aplicaciones que componen el proyecto.

- Por otra parte, tenemos los directorios **accounts/** y **tasks/**, que corresponden a las 2 aplicaciones que forman nuestro proyecto. La aplicación Accounts se encarga de toda la parte relacionada con las cuentas de usuario. Mientras que la aplicación Tasks se encarga de la parte de evaluación de tareas y progreso del camino de vida.

  Dentro de estos directorios se almacenan los ficheros que contienen los modelos, vistas y *templates* correspondientes a cada aplicación; así como los directorios que contienen los ficheros estáticos de cada una.

- Por último, tenemos el directorio **media/**, en donde se almacenan las imágenes personales subidas por los usuarios a su perfil.

Tras la creación del proyecto, procedimos a la configuración de acceso a la base de datos de PostgreSQL. Por defecto los proyectos de Django utilizan SQLite como motor de la base de datos, por lo tanto, realizamos cambios en el fichero *settings.py*, en donde indicamos que PostgreSQL sería el motor de la base de datos usada, además de indicar las credenciales de acceso:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'caminodevidaDB',
        'USER': 'DjangoUserDB',
        'PASSWORD': '********',
    }
}
```

Llegados a este punto, ya disponemos de todas las herramientas correctamente configuradas para desarrollar el código de la aplicación.

El primer paso fue desarrollar la aplicación *Accounts*, para así implementar las funcionalidades de registro, acceso, perfil de usuario, cambio de pin, elegir preferencias de accesibilidad y subida de imágenes personales. Para parte de esta aplicación se decidió utilizar el *middleware 'django.contrib.auth.middleware'* , el cual nos proporciona modelos con campos básicos para los usuarios, además de implementar la lógica correspondiente al registro, acceso, validaciones de los datos introducidos por los usuarios y algunas medidas de seguridad. Para realizar esto usamos como referencia la implementación realizada en el libro *Django 2 by Example* [21].

En nuestro caso decidimos reutilizar el campo *nombre de usuario* y *password* del modelo básico que nos suministra, que se corresponden con las credencias de acceso de los alumnos. Pero como además necesitamos almacenar información adicional de los usuarios, definimos un modelo denominado *Profile*, el cual tiene una relación *OneToOneField* con el modelo de Usuarios que nos proporciona el *middleware*; este modelo añade los campos necesarios para almacenar el nombre completo, fecha de nacimiento, género, fotos personales y preferencias de accesibilidad de los usuarios.

Una vez se completó definición del modelo, se realizó la migración de este a la base de datos. Para posteriormente desarrollar los *templates*, vistas y formularios necesarios para proporcionar las funcionalidades antes mencionadas. Además de definir las rutas URL correspondientes a cada pantalla. Durante esta parte del desarrollo, también se aprovechó para crear un *template* que sirva de base para todas las pantallas de la aplicación, así como una hoja de estilos CSS con las clases por defecto de la aplicación.

Tras esto, se realizó una serie de pruebas básicas para comprobar el correcto funcionamiento de cada una de las funcionalidades implementadas en *Accounts*, y así poder detectar errores de programación y corregirlos.

Posteriormente, pasamos al desarrollo de la aplicación *Tasks*, que incorpora las funcionales de ver lista de tareas, evaluar tareas, ver progreso del camino de vida e imprimir la lista de tareas y el progreso.

De igual manera que en la primera aplicación, comenzamos definiendo los modelos necesarios para almacenar la información, por una parte, tenemos el modelo *Group* que establece que los objetos de este modelo tendrán un título, una categoría y un estatus; y que tendrán como como clave foránea la entidad Usuario. Mencionar que el atributo estatus se utiliza para implementar la funcionalidad de activar o desactivar determinados grupos de contenido.

También definimos el modelo *Task* que establece que las tareas tendrán un título, una evaluación y el nombre del pictograma asociado a cada tarea; en este caso se establece que la clave foránea será la entidad *Group*. Destacar que también se indica la lista de valores (rojo, amarillo y verde) válidos para el campo evaluación, que como hemos mencionado en capítulos anteriores, se utiliza un código de colores basado en los semáforos.

Tras definir los modelos, realizamos la migración correspondiente para que sean aplicados en la base de datos. Y así poder empezar a implementar las funcionalidades de esta aplicación.

En primer lugar, se desarrolló la pantalla de inicio con su correspondiente *template*, vista y definición de la URL. Como esta pantalla utiliza pictogramas como ficheros estáticos, se requirió una pequeña modificación en la configuración del proyecto para que el servidor de Django pueda servir ficheros estáticos, básicamente hubo que indicar la ruta de los ficheros estáticos en el fichero *settings.py* e incluir un *helper* en el fichero *urls.py* del proyecto.

Posteriormente se procedió a implementar la funcionalidad de autoevaluación de tareas, es decir, desarrollando su correspondiente *template*, formularios, vista y definición de la URL. En está pantalla se decidió realizar una paginación de la lista de tareas según la categoría a la que corresponden, para de esta manera simplificar la navegación en la pantalla. Destacar que en este punto fue necesario realizar una modificación en la aplicación anterior, concretamente en la vista de registro de usuarios, donde fue necesario añadir la instanciación de toda la lista de tareas y grupos de un usuario al completar el registro correctamente.

En la siguiente etapa, se implementó la funcionalidad de consultar el progreso del camino de vida; desarrollando de igual manera sus vistas, *templates* y formularios. El componente que más trabajo llevó, fue la vista, dado que hubo que definir todas las condiciones que determinan cuando se ha completado cada meta del camino de vida, el proceso consiste en obtener la lista de tareas del usuario e ir comprobando si el usuario es capaz de realizar todas las tareas en cada meta; con toda esta información se crea un diccionario que incluye la información para representar cada meta: la lista de tareas, su estado (completada, a medias o incompleta) y el nombre de su pictograma correspondiente. Con ese diccionario y obteniendo las imágenes personales del usuario, posteriormente el *template* genera la pantalla.

En último lugar, se trabajó para implementar las funcionalidades de poder imprimir la lista de tareas y progreso del camino de vida. Como sabemos los navegadores web incluyen una funcionalidad para poder imprimir cualquier página web, pero para que estas se impriman correctamente fue necesario trabajar en una hoja de estilos CSS que se aplica cuando se abre el diálogo de impresión.

Para ello creamos el fichero *print.css* en donde se definieron algunas reglas con el atributo *@media print*, que básicamente define unos márgenes y evita que haya saltos de página en medio de los elementos; además para que esto funcione correctamente fue necesario añadir el atributo HTML *media="print"*, a la hora de incluir el fichero *print.css* en los *templates* de estas dos páginas a imprimir.

Mencionar que, como recurso de consulta adicional, en cuanto a detalles de implementación de las vistas, *templates*, formularios y modelos, se ha usado el libro *Beginning Django* [22].

El resultado de esta fase de desarrollo se puede observar en las capturas de pantalla de la aplicación que se incluyen en el **Anexo B**.

# 7 Integración, Pruebas y resultados

En esta sección se describe el alcance que cubren las pruebas ejecutadas. Así como el conjunto de pruebas realizadas a la aplicación, con el resultado obtenido en cada una de ellas.

## *7.1 Integración*

Para llevar a cabo la integración de la aplicación se han seguido los pasos explicados en el Manual de instalación (véase el **Anexo C**). Para la elaboración de dicho manual se ha usado como referencia un artículo publicado en *digitalocean* [23], así como la información de la documentación oficial de Django acerca de cómo servir ficheros [24].

Tras terminar el despliegue de la aplicación en el portátil que hará de servidor, se acordó una reunión con los responsables de Alenta, en donde se realizó una demostración de las funcionalidades de la aplicación. Así como proporcionarles las credenciales de administrador y unas instrucciones de cómo llevar a cabo el arranque del servidor.

En la demostración de la aplicación, simulamos el uso que haría un nuevo usuario, desde un navegador web accedemos a la aplicación usando la IP local del servidor, esto nos dirige a la pantalla de acceso, como no tenemos cuenta pulsamos en el botón 'Regístrate'.
En la pantalla de registro rellenamos el formulario con todos nuestros datos personales y confirmamos el registro, tras esto la aplicación nos lleva de nuevo a la pantalla de acceso, en donde introducimos nuestras credenciales y pulsamos el botón 'Acceder'. Ya en la pantalla de inicio, vamos a Configuración en donde desactivamos los grupos de tareas que no queremos trabajar, y definimos las opciones de accesibilidad a nuestro gusto.

Tras esto volvemos a la pantalla de inicio y nos dirigimos a la pantalla de Autoevaluación, en donde podemos ver el listado de tareas paginadas según la categoría del grupo, buscamos unas tareas, las valoramos y confirmamos el cambio para guardar nuestra habilidad en dichas tareas del ciclo de vida. Además, en esta pantalla tenemos la posibilidad de generar un documento para imprimir con la lista de tareas, pulsando en 'Imprimir'.

Luego mostramos la parte de visualizar el progreso del camino de vida, para ello desde la pantalla de inicio pulsamos en el pictograma de 'Ver camino de vida', con lo que accedemos a la pantalla de progreso del camino de vida, en donde podemos ver las metas que están marcadas en el ciclo de vida de los alumnos, además de identificar el avance de sus habilidades en una meta concreta, fijándonos en el color del borde del pictograma y en la lista de tareas de cada meta. Esta pantalla es personalizable, podemos añadir fotos al inicio o final de cada etapa, para ello nos desplazamos al Perfil de usuario y seleccionamos el enlace para subir imágenes, en dicha sección elegimos la etapa correspondiente a la foto que vamos a subir, buscamos la foto en el dispositivo y confirmamos los cambios; tras esto volvemos al progreso del camino de vida, en donde ya figuran las imágenes personales que acabamos de subir. Esta pantalla también de posibilidad de generar un documento para imprimir.

También se les mostró como acceder al panel del administrador, en donde pueden listar los usuarios, y restablecer la contraseña en caso de que un alumno la haya olvidado.

Los responsables de Alenta mostraron su satisfacción en cuanto a la aplicación desarrollada, ya que consideran que se han cumplido las necesidades que mostraron a lo largo de las reuniones anteriores, y por lo tanto que les va ser de gran ayuda para realizar el seguimiento del ciclo de vida de los alumnos.

## *7.2 Alcance de las pruebas*

El tipo de pruebas que se ha escogido realizar han sido las pruebas funcionales, con el objetivo de comprobar y validar el correcto funcionamiento de las funcionalidades que se han implementado en la aplicación y que figuraban en la especificación de requisitos.

Las pruebas se han ido ejecutando a lo largo del desarrollo de la aplicación, en pequeñas etapas de validación, así como una vez finalizado el proceso de despliegue.

## *7.3 Conjunto de pruebas realizadas*

En este subapartado se muestra el conjunto de pruebas funcionales realizadas a la aplicación para verificar el cumplimiento de los requisitos funcionales especificados durante la fase de análisis:

**Prueba RF1 [Realizada con éxito]**
Esta prueba verifica que los nuevos usuarios se puedan registrar correctamente en la aplicación.

Para realizarla se accede a la pantalla de registro y se introducen los datos del usuario (nombre completo, fecha de nacimiento, género, usuario y pin), tras ello se confirma el registro con el botón correspondiente.

El resultado esperado es la visualización del mensaje de usuario registrado correctamente, así como la correcta inserción en la base de datos de la información introducida por el usuario.

**Prueba RF2.1 [Realizada con éxito]**
Esta prueba verifica el correcto acceso de los usuarios con las credenciales introducidas por ellos durante la fase de registro.

Para realizarla se accede a la pantalla de acceso y se introducen las credencias del usuario en cuestión.

El resultado esperado es el acceso a la pantalla de inicio de la aplicación tras haber comprobado la validez de las credenciales introducidas.

**Prueba RF2.2 [Realizada con éxito]**
Esta prueba verifica el correcto acceso del administrador al panel de administrador usando las credenciales proporcionadas por el desarrollador.

Para ello se accede a la pantalla de acceso de administradores y se introducen las credenciales de su cuenta.

El resultado esperado en este caso, es el acceso al panel de administrador, donde puede ver la lista de usuarios registrados, y realizar sus funciones correspondientes.

**Prueba RF3 [Realizada con éxito]**
Con esta prueba se verifica que los usuarios pueden consultar sus datos personales en la sección de perfil de usuarios.

Para realizarla, se realiza el paso previo de acceso a la aplicación, y tras ello se transita al perfil de usuario mediante el enlace disponible en la barra de navegación de la aplicación.

El resultado esperado es acceder al perfil de usuario, donde se visualizarán los datos que el usuario introdujo durante su registro.

**Prueba RF4 [Realizada con éxito]**
Esta prueba verifica que los usuarios puedan cambiar su pin/contraseña de acceso a la aplicación cuando lo consideren oportuno.

Para realizarlo es necesario haber accedido previamente, tras esto se navega hasta la pantalla del perfil de usuario y se pulsa el enlace para acceder a la pantalla de cambio de pin. En dicha pantalla se introduce el pin actual, el nuevo pin y la confirmación del pin nuevo, además de confirmar el cambio con el botón correspondiente.

El resultado esperado es la visualización del mensaje de cambio de pin correcto, así como la modificación en la base de datos del pin de acceso del usuario.

**Prueba RF5 [Realizada con éxito]**
Esta prueba garantiza que los usuarios pueden subir las imágenes personales correspondientes a las distintas etapas de su camino de vida.

Para realizarla es necesario haber accedido previamente, posteriormente desde su perfil de usuarios se accede a la pantalla de subida de imágenes, se pulsa en el botón para seleccionar una imagen de la etapa deseada y se confirma la subida.

El resultado esperado es el almacenamiento de la imagen en la aplicación en el directorio */media*, para luego ser visualizada en la pantalla de progreso del camino de vida.

**Prueba RF6 [Realizada con éxito]**
Con esta prueba se verifica la correcta configuración de los contenidos que muestra la aplicación en la pantalla de autoevaluación.

Para realizarla se accede previamente a la aplicación, tras ello se pulsa en el enlace de configuración en la barra de tareas y se marcan o desmarcan los grupos de contenidos según se quieran activar o desactivar, y se confirma pulsando en el botón de confirmación.

El resultado esperado es que la aplicación solo muestre los contenidos correspondientes a los grupos que el usuario ha dejado activados, y por lo tanto que la variable *status* de cada grupo tenga en la base de datos el valor indicado por el usuario.

**Prueba RF7 [Realizada con éxito]**
Esta prueba valida que un usuario pueda listar sus grupos de tareas, donde podrá visualizar la información de cada una (pictograma, título y evaluación).

Para realizarla en primer lugar se accede a la aplicación, y después desde la pantalla de inicio se pulsa en el pictograma correspondiente a la autoevaluación de tareas.

El resultado en esta prueba es que el usuario visualice un listado con las tareas que puede evaluar y que tiene activadas.

**Prueba RF8 [Realizada con éxito]**
Esta prueba verifica que el usuario puede autoevaluar una tarea, para así ir registrando su progreso en la adquisición de habilidades a lo largo de su desarrollo.

Para realizarla se accede previamente a la aplicación y se transita a la pantalla de autoevaluación de tareas, en donde se valora una tarea seleccionando el color correspondiente y se confirma pulsando en el botón de confirmar cambios.

El resultado esperado es que la valoración seleccionada se muestra correctamente, así como la modificación en la base de datos del campo *evaluation* de las tareas respectivas.

**Prueba RF9 [Realizada con éxito]**
Esta prueba valida que los usuarios puedan consultar el progreso de su camino de vida, de acuerdo a las valoraciones que hayan hecho.

Para realizarla primero se accede a la aplicación y en la pantalla de inicio se pulsa en el pictograma ‘Ver camino de vida’.

El resultado esperado es visualizar la pantalla del progreso del camino de vida, en donde se puede comprobar el estado de cada meta, que se representa con colores en el borde de los pictogramas. Además de ver sus imágenes personales en el comienzo o final de cada etapa.

**Prueba RF10 [Realizada con éxito]**
Esta prueba verifica que un alumno pueda visualizar información de como completar una determinada meta del camino de vida.

Para realizarla se accede a la aplicación y se navega hasta la pantalla de progreso del camino de vida, después se pulsa en el botón para ver la lista de tareas de la meta deseada.

El resultado esperado es ver una ventana emergente con la lista de tareas que se deben realizar para dar por completada una meta, destacar que las tareas ya realizadas se muestran en verde.

**Prueba RF11 [Realizada con éxito]**
Con esta prueba se verifica que un usuario pueda imprimir la lista completa de tareas, con toda la información asociada a cada tarea.

Para realizarla se accede a la aplicación y se transita hasta la pantalla de autoevaluación, en dicha pantalla se pulsa el botón imprimir.
El resultado esperado es que se abra una nueva ventana con el formato para imprimir y cuando la página esté completamente cargada se muestre automáticamente el diálogo del navegador para imprimir.

**Prueba RF12 [Realizada con éxito]**
Con esta prueba se valida que el administrador pueda dar de baja la cuenta de un usuario.

Para realizarla el administrador se debe autenticar en su pantalla de acceso, tras ello en su panel de inicio selecciona la categoría Usuarios para ver la lista de alumnos registrados en la aplicación. En dicho listado selecciona la cuenta que quiere dar de baja y confirma la acción.

El resultado esperado es que la cuenta seleccionada sea dada de baja correctamente, y por lo tanto no se pueda acceder con las credenciales del usuario desactivado. En el caso de que la baja sea definitivamente, se borran los datos del usuario.

# 8 Conclusiones y trabajo futuro

## *8.1 Conclusiones*

En este Trabajo de Fin de Grado hemos llevado a cabo el desarrollo de una aplicación web orientada a realizar el seguimiento en la adquisición de habilidades por parte de personas con trastornos del espectro autista. Para ello hemos seguido un el ciclo de vida en cascada con retroalimentación, pasando por todas las fases, exceptuando la de mantenimiento, que por razones obvias quedará especificada como trabajo futuro.

Durante el desarrollo hemos tenido una serie de reuniones con los responsables de Alenta, por lo que se podría decir que ha sido un proceso interactivo con los clientes, ya que les hemos ido presentando los resultados de cada fase para que nos muestren su opinión y así poder ir ajustando la aplicación a sus necesidades. Como conclusión a esto, podemos decir se ha seguido una línea de trabajo con cierta similitud a las que se desarrollan entornos laborales, lo cual marca una diferencia respecto a otros trabajos de desarrollo de aplicaciones.

En cuanto a la aplicación, podemos concluir que, de acuerdo a realimentación que nos dieron los clientes al realizar la integración, se han implementado las funcionalidades requeridas para cubrir el problema tal y como lo hacía la solución utilizada hasta el momento. Además de incluir algunas mejoras, como por ejemplo las opciones de accesibilidad, para adaptarla a las necesidades específicas que requieren el tipo de usuarios a los que va dirigida.

Para concluir, en relación a los contenidos aprendidos, podemos destacar que se ha profundizado el aprendizaje del *framework* Django, el cual fue utilizado en la asignatura de Proyectos de Sistemas Informáticos. También he aprendido a usar la herramienta Boostrap que ha sido empleada para la parte de desarrollo *front-end*; además del proceso de despliegue de una aplicación web, aunque en este caso se ha desplegado en una red local.

## *8.2 Trabajo futuro*

La principal línea de trabajo que se puede seguir a partir de la aplicación desarrollada es trabajar en medidas para garantizar un tratamiento seguro de los datos personales e imágenes de los usuarios cumpliendo con los requisitos marcados por la Ley Orgánica de Protección de Datos de Carácter Personal, para así poder llevar a cabo un despliegue en una red pública como es Internet, de tal manera que la aplicación sea accesible por usuarios externos, no solamente por los alumnos de la asociación Alenta.

Por otra parte, como sabemos el resultado de este TFG es una primera versión de la aplicación, por lo tanto, está perfectamente sujeta a seguir mejorándose en una etapa de mantenimiento o bien volviendo a iterar sobre todas las fases del ciclo de vida en caso de que sean cambios muy significativos.

Por ejemplo, se puede trabajar en incluir más opciones de accesibilidad para mejorar su uso por parte de este tipo de usuarios, como podría ser la integración de un sintetizador de voz para escuchar determinados textos que muestra la aplicación. O bien en incluir cualquier otro tipo de funcionalidad adicional que requiera el cliente.

# Referencias


[1] “MyLifePlan”, https://apps.apple.com/us/app/mylifeplan/id1103504949, accedido: 11/02/2019

[2] “GoalPlanDo”, https://apps.apple.com/us/app/goalplando/id961194974, accedido: 11/02/2019

[3] “My Supports by MyKey™ Consulting Services”, https://apps.apple.com/us/app/my-supports-by-mykey-consulting-services/id1137646050, accedido: 11/02/2019

[4] “AutiPlan”, https://autiplan.com/, accedido: 28/03/2019

[5] Dr. Daniel Tapias, Proyectos de Desarrollo Software, http://arantxa.ii.uam.es/~proyectos/teoria/C5_Proyectos%20de%20desarrollo%20software.pdf, accedido: 04/01/2019

[6] “Django”, https://www.djangoproject.com/, accedido: 10/01/2019

[7] “Laravel”, https://laravel.com/, accedido: 11/01/2019

[8] “Web Framework Benchmarks”, http://www.techempower.com/benchmarks/#section=data-r16&hw=ph&test=json, accedido: 11/02/2019

[9] “PostgreSQL”, https://www.postgresql.org/, accedido: 20/01/2019

[10] “MySQL”, https://www.mysql.com/, accedido: 20/01/2019

[11] “SQLite3”, https://www.sqlite.org/index.html, accedido: 20/01/2019

[12] “Oracle”, https://www.oracle.com/es/database/, accedido: 21/01/2019

[13] “Bootstrap”, https://getbootstrap.com/docs/4.3/getting-started/introduction/, accedido: 26/01/2019

[14] “Semantic UI”, https://semantic-ui.com/, accedido: 27/01/2019

[15] D. Javier Jesús Gutiérrez Rodríguez, “Diagramas UML de casos de uso y de requisitos Diagramas UML de casos de uso y de requisitos”, http://www.lsi.us.es/~javierj/cursos_ficheros/metricaUML/CasosUsoUML.pdf

[16] “Modelo Vista Controlador”, https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html, accedido: 05/02/2019

[17] “Modelo Vista Template”, https://docs.djangoproject.com/es/2.2/faq/general/, accedido: 05/02/2019

[18] “Diseño de bases de datos”, http://arantxa.ii.uam.es/~rcobos/teaching/esp/si1/tema2_fin.pdf, accedido: 07/02/2019

[19] “Balsamiq Mockups 3”, https://balsamiq.com/, accedido: 12/02/2019

[20] “Python Package Index”, https://pypi.org/, accedido: 10/06/2019

[21] Antonio Mele, “Django 2 by Example: Build powerful and reliable Python web applications from scratch”, Packt Publishing, mayo de 2018

[22] Daniel Rubio, “Beginning Django Web Application Development and Deployment with Python”, Apress, 2017

[23] Justin Ellingwood, “How To Serve Django Applications with Apache and mod_wsgi on Ubuntu 14.04”, https://www.digitalocean.com/community/tutorials/how-to-serve-django-applications-with-apache-and-mod_wsgi-on-ubuntu-14-04, accedido: 02/06/2019

[24] “How to use Django with Apache and mod_wsgi”, https://docs.djangoproject.com/en/2.1/howto/deployment/wsgi/modwsgi/#serving-files, accedido: 03/06/2019

# Glosario

| | |
|---|---|
| API | Application Programming Interface |
| CSS | Cascading Style Sheets |
| HTML | HyperText Markup Language |
| SQL | Structured Query Language |
| MVT | Model View Template |
| PyPI | Python Package Index |
| IP | Internet Protocol |

# Anexos

## *A. Maquetas interfaz gráfica*

En este anexo se incluyen las maquetas de las pantallas que formarán la aplicación Camino de Vida:


**Figura 0-1: Maqueta de pantalla de inicio de sesión**


**Figura 0-2: Maqueta de pantalla de registro de usuarios**

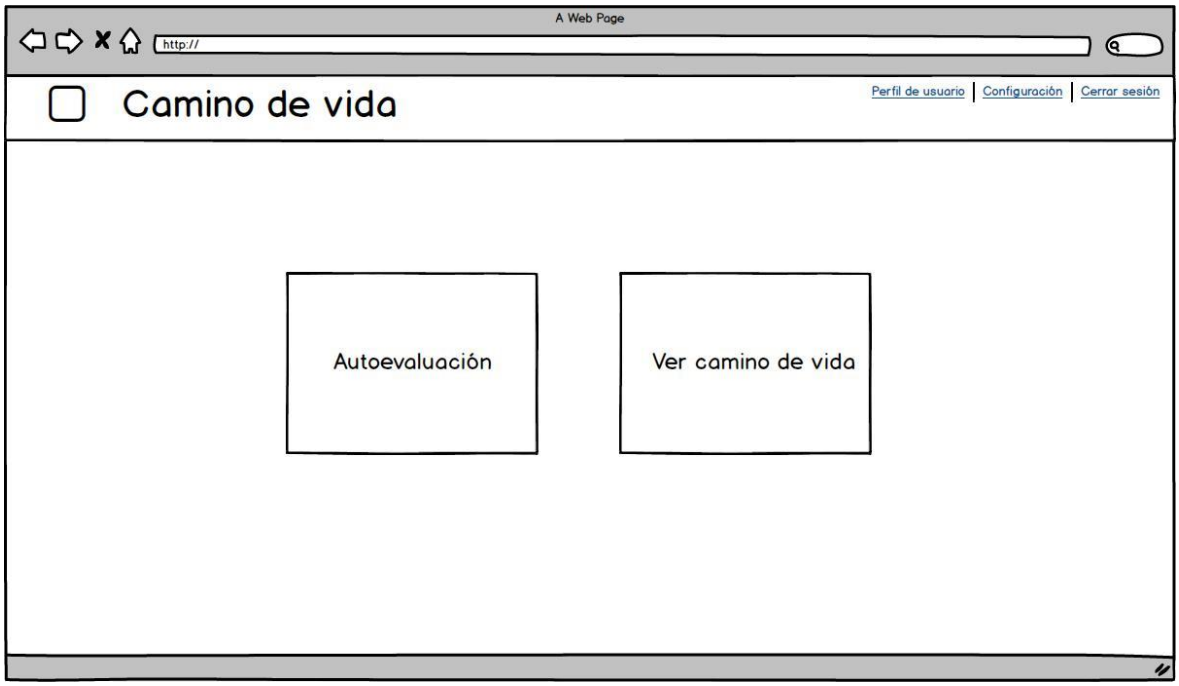


**Figura 0-3: Maqueta de pantalla de inicio**

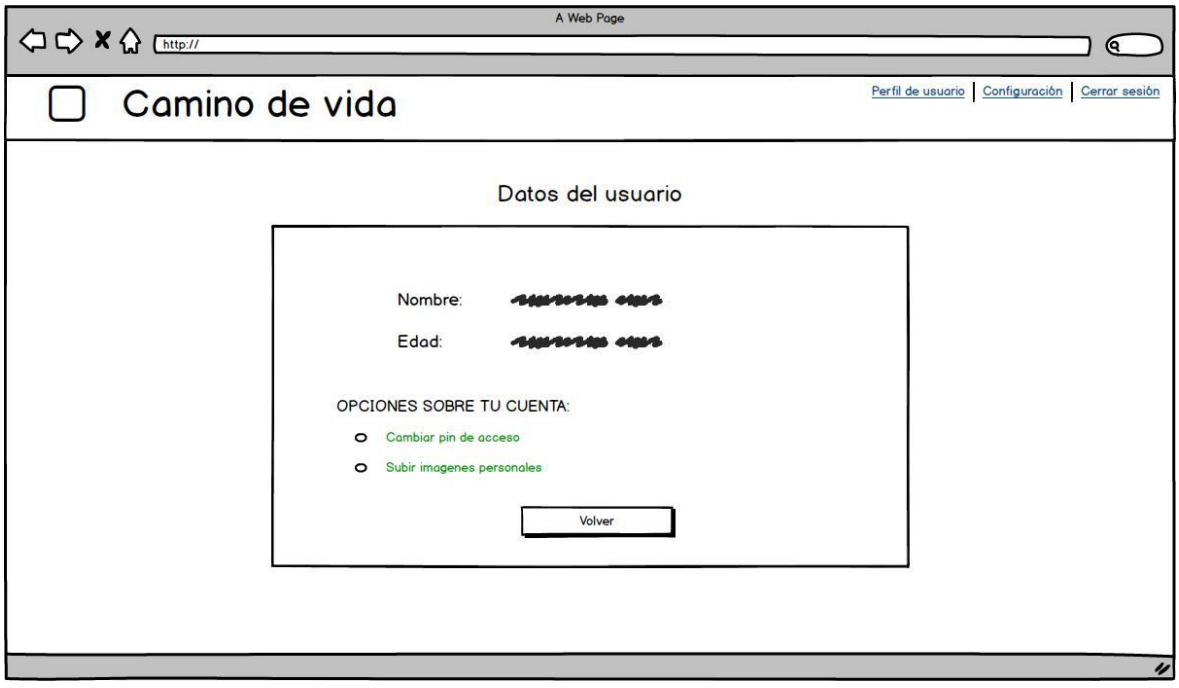


**Figura 0-4: Maqueta de pantalla de perfil de usuario**

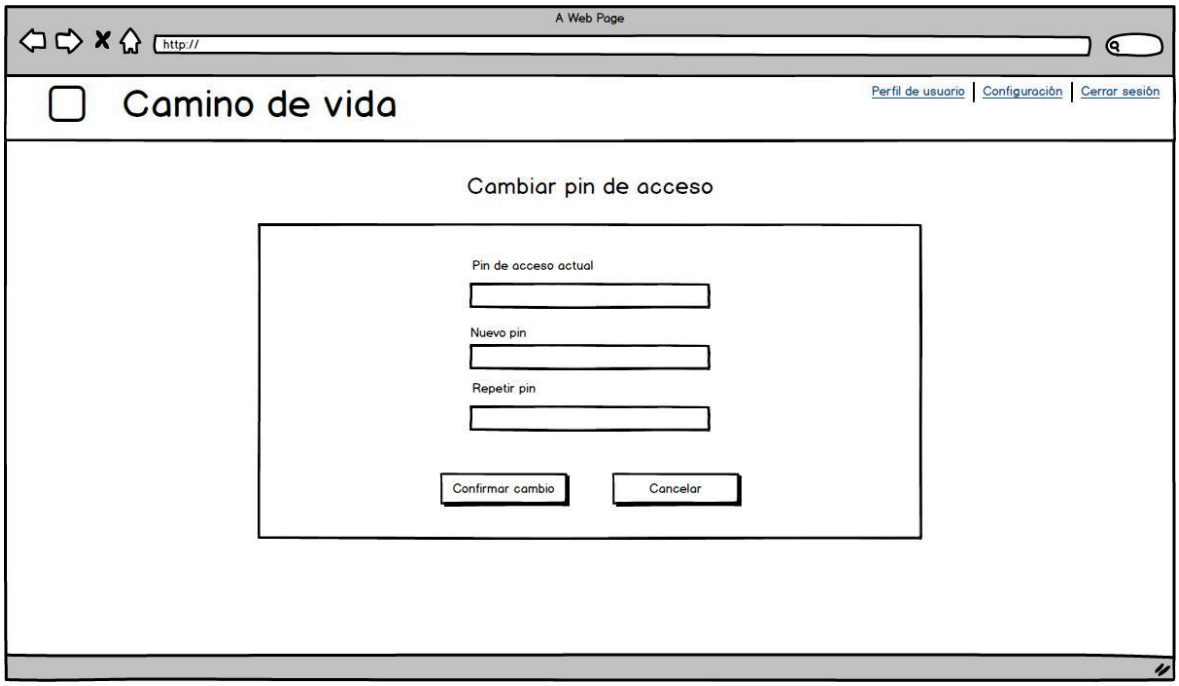


**Figura 0-5: Maqueta de pantalla de cambio del pin de acceso**

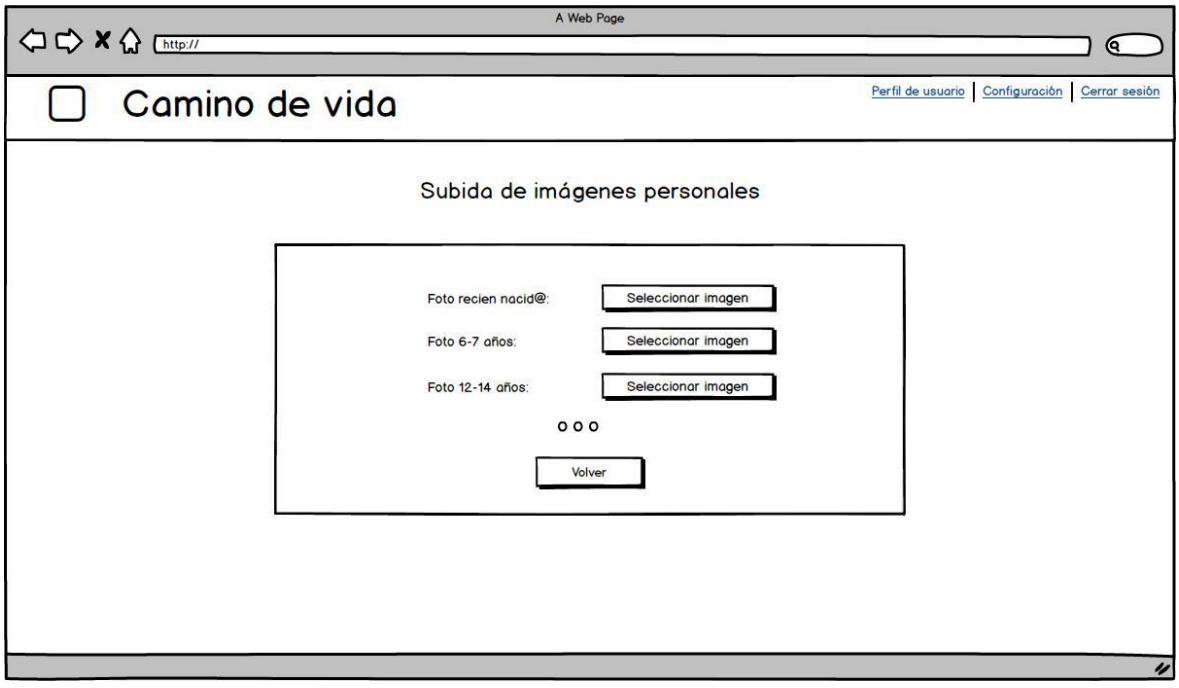


**Figura 0-6: Maqueta de pantalla de subidad de imágenes personales**

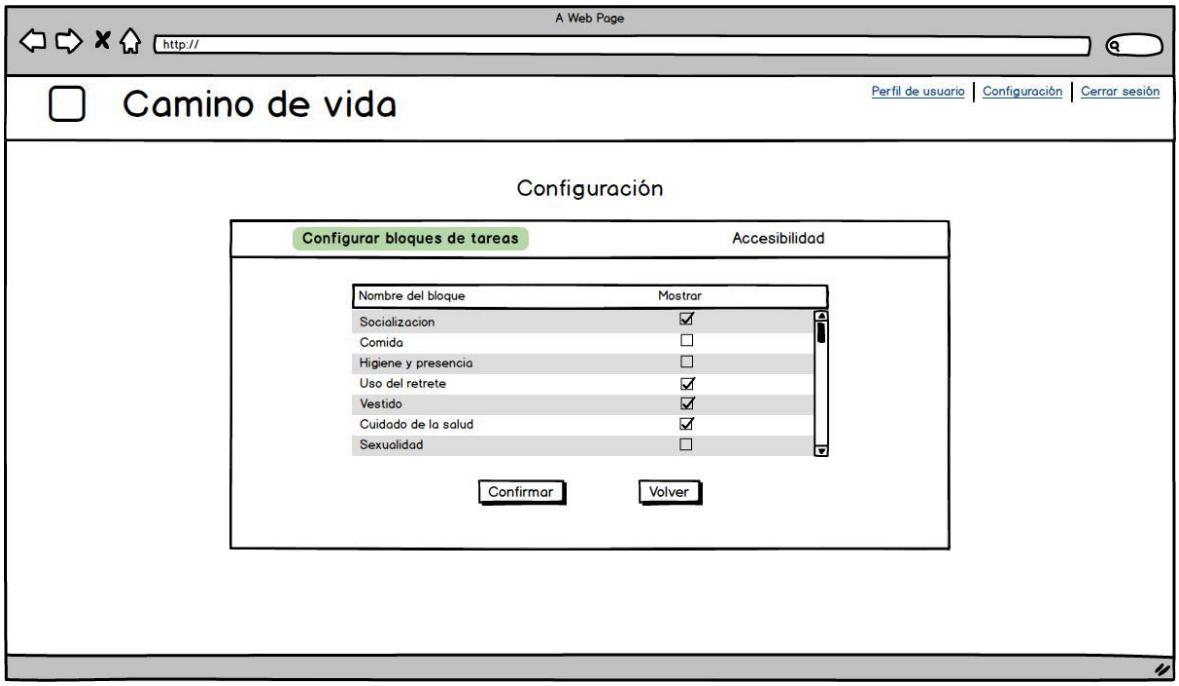


**Figura 0-7: Maqueta de pantalla de configuración de contenidos**

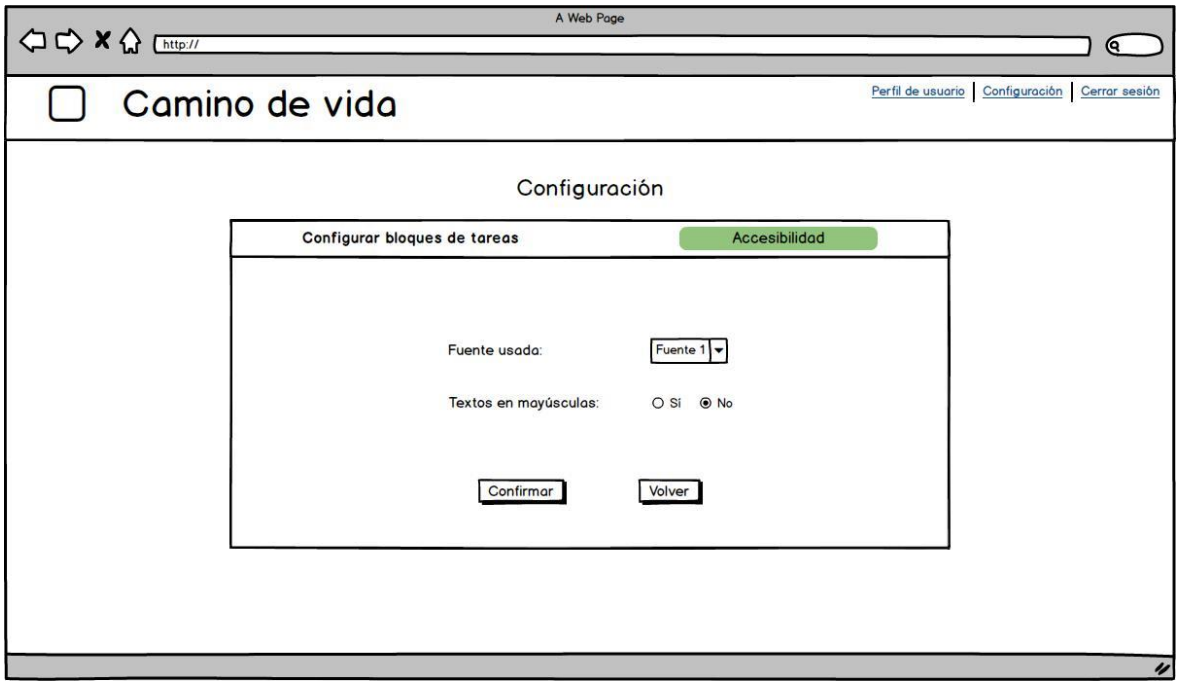


**Figura 0-8: Maqueta de pantalla de configuración de opciones de accesibilidad**

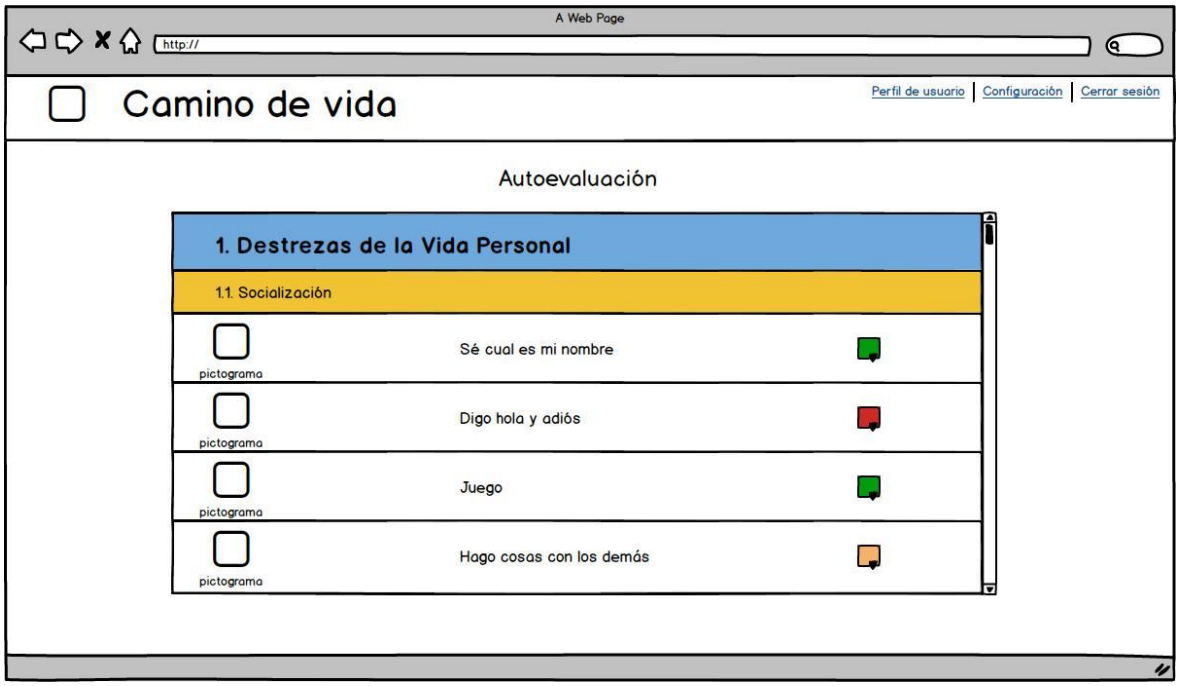


**Figura 0-9: Maqueta de pantalla de autoevaluación de tareas**

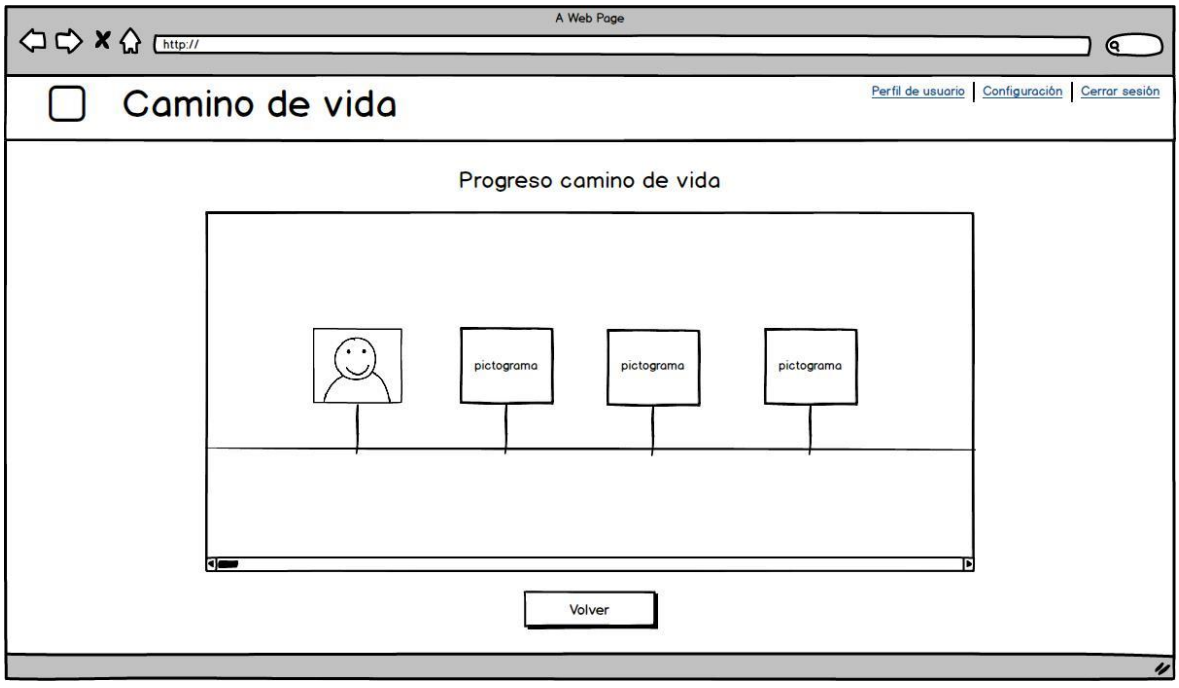


**Figura 0-10: Maqueta de pantalla de progreso del camino de vida**

## *B. Capturas de pantalla de la aplicación*

En este anexo se incluyen capturas de pantalla de todas las secciones que conforman la aplicación desarrollada a lo largo del TFG:

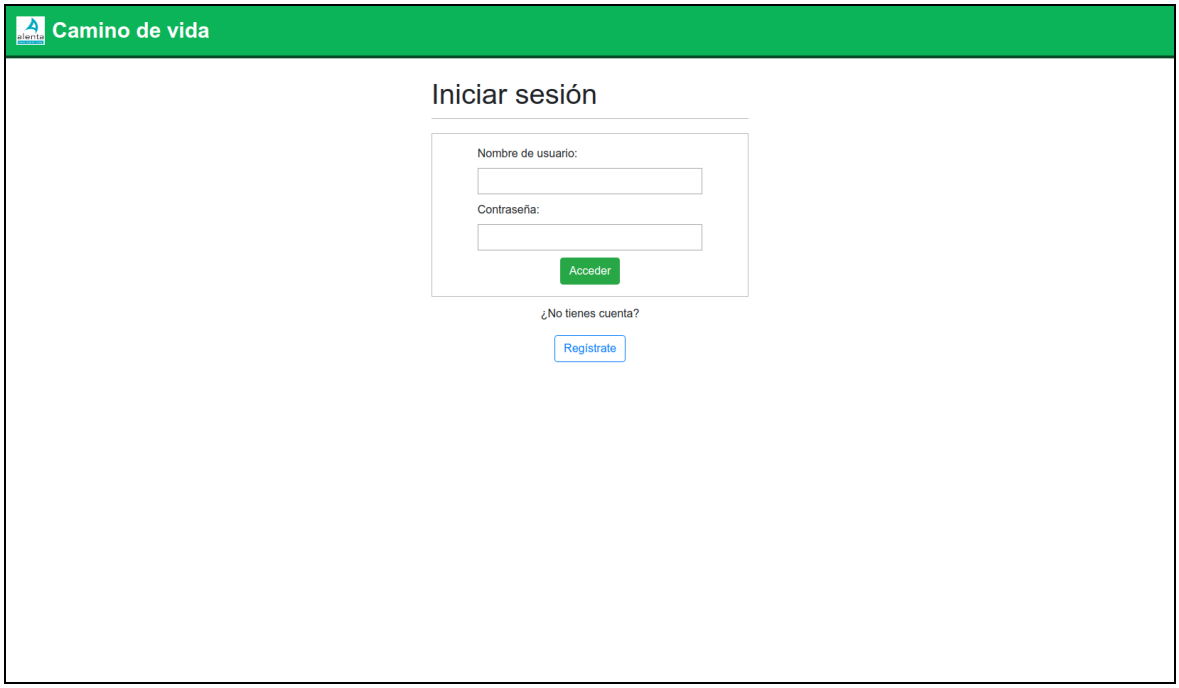


**Figura 0-11: Captura de pantalla de inicio de sesión**

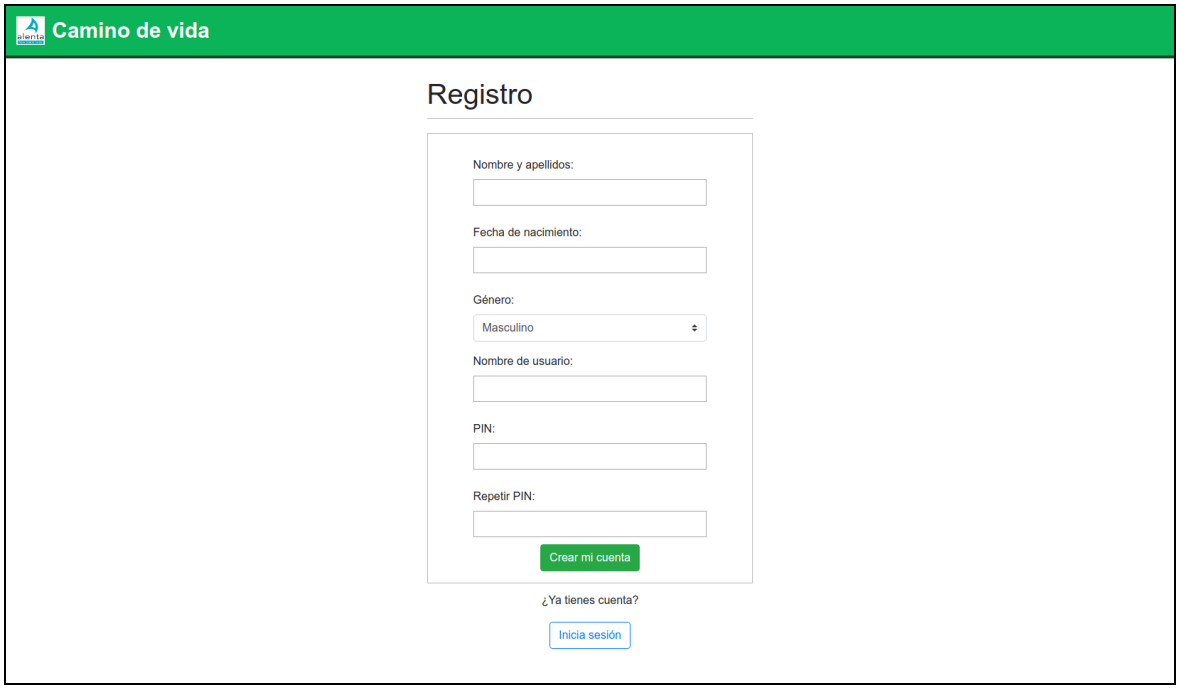


**Figura 0-12: Captura de pantalla de registro de usuarios**

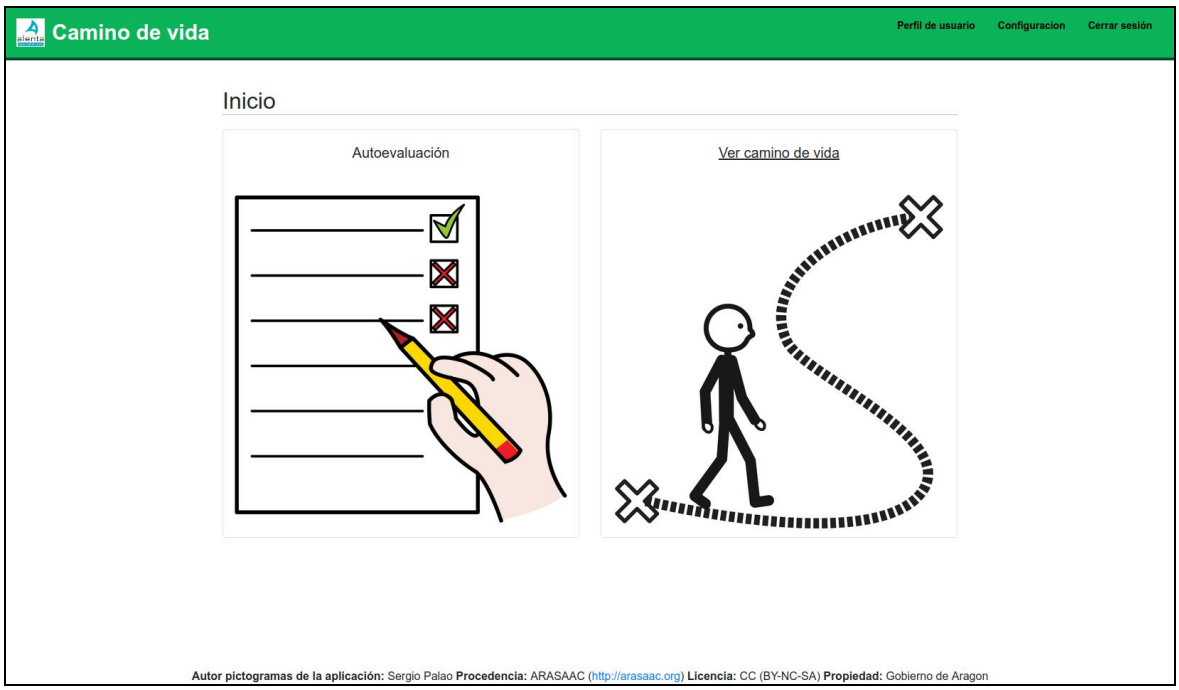


**Figura 0-13: Captura de pantalla de inicio**

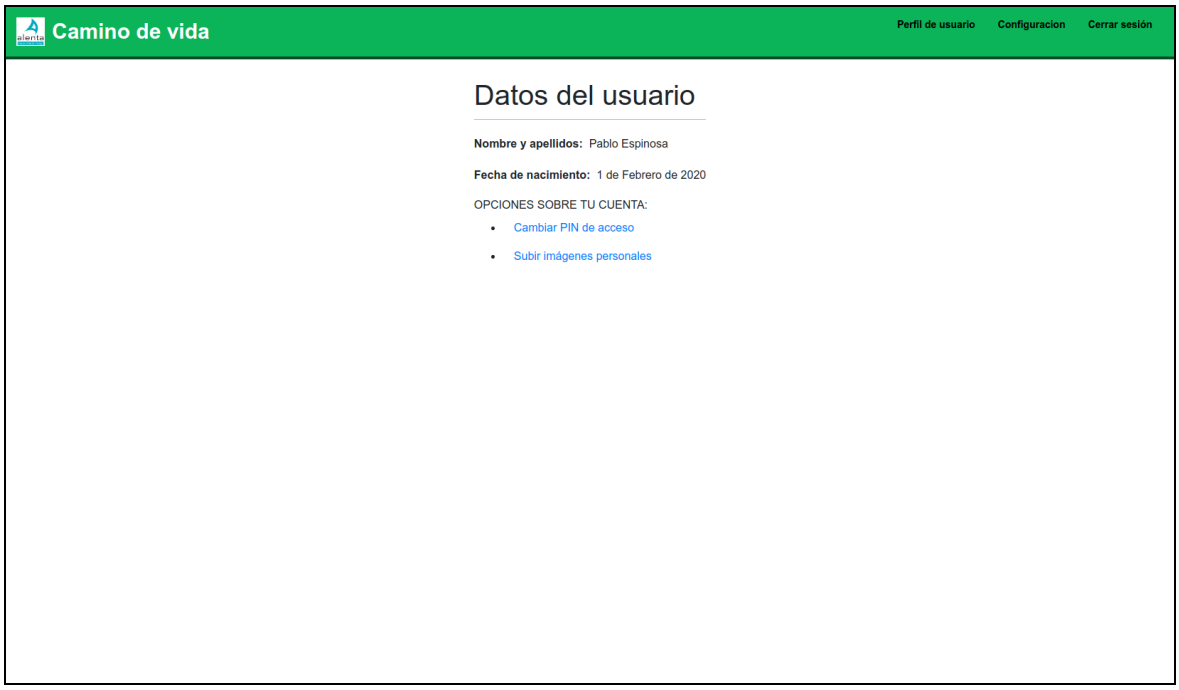


**Figura 0-14: Captura de pantalla de perfil de usuario**

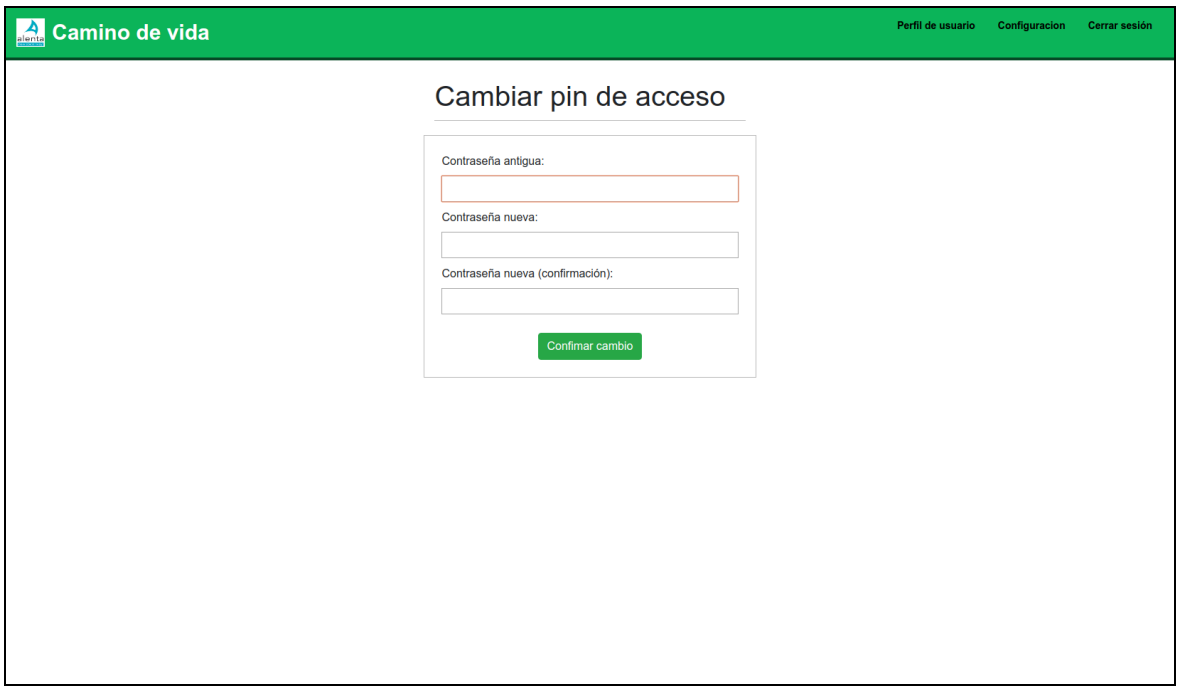


**Figura 0-15: Captura de pantalla de cambio de contraseña**


**Figura 0-16: Captura de pantalla de subida de imágenes**

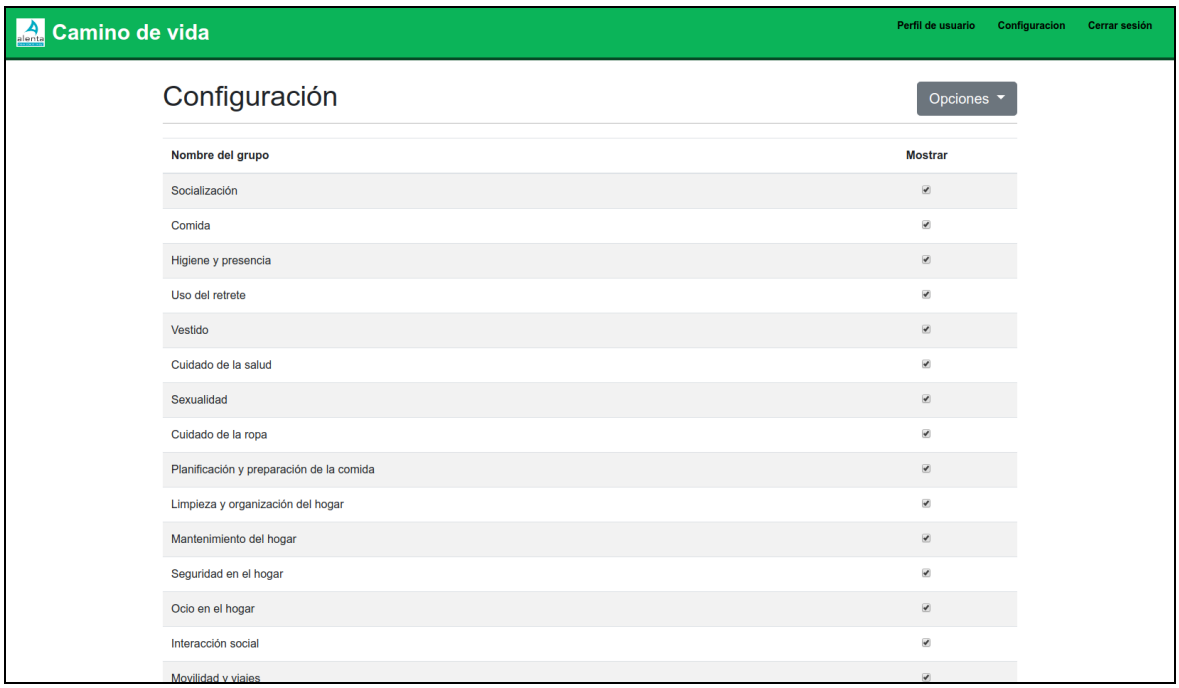


**Figura 0-17: Captura de pantalla de configuración de contenidos**

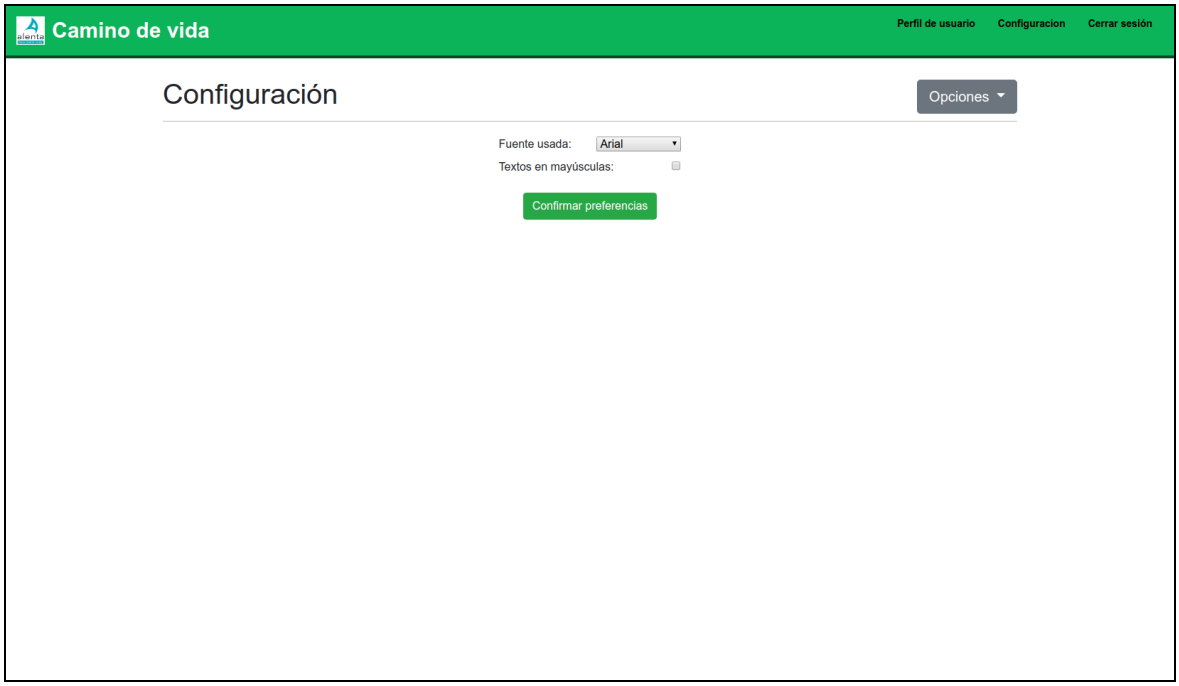


**Figura 0-18: Captura de pantalla de configuración de opciones de accesibilidad**

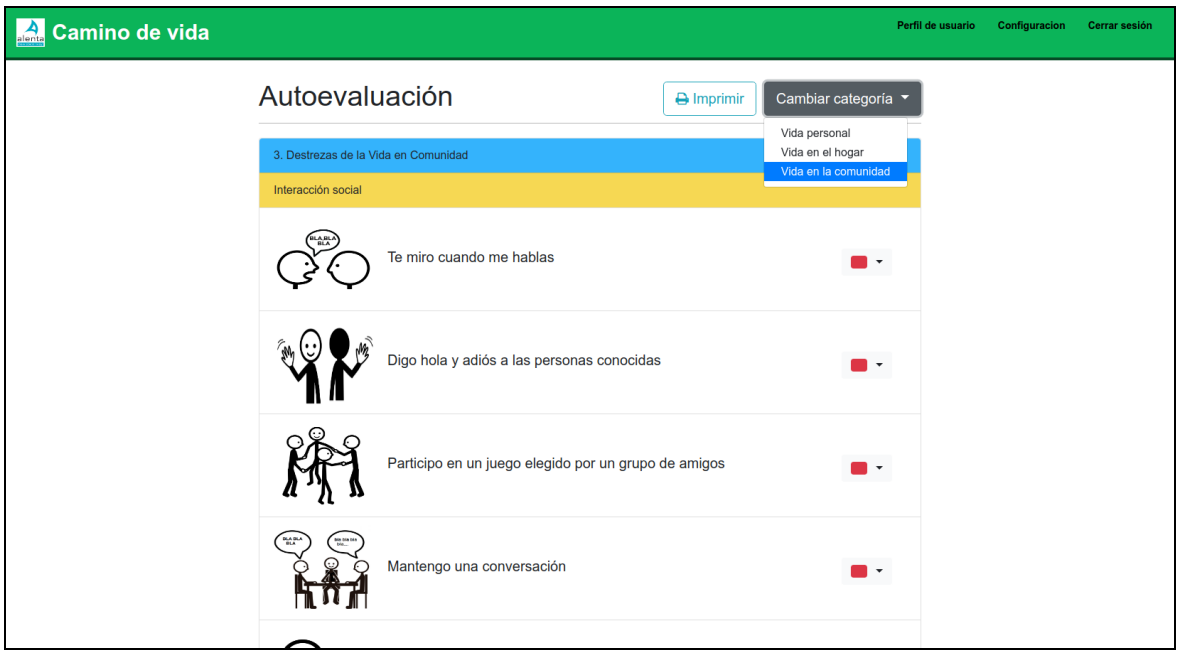


**Figura 0-19: Captura de pantalla de autoevaluación de tareas**

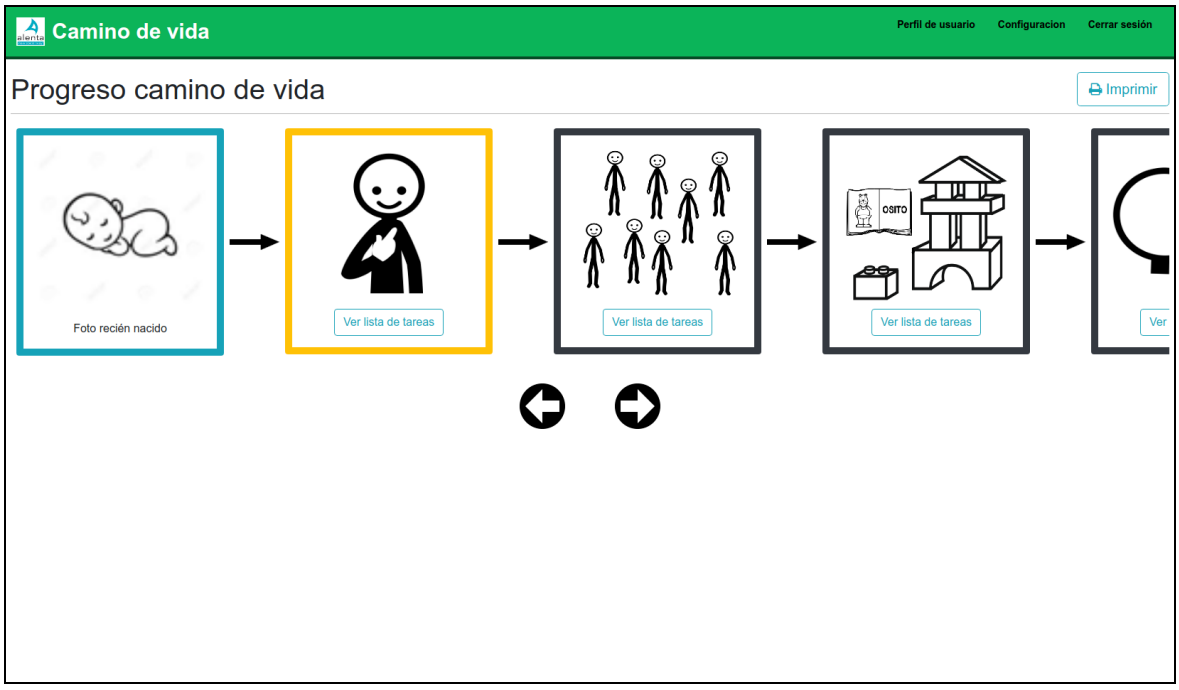


**Figura 0-20: Captura de pantalla del progreso del camino de vida**

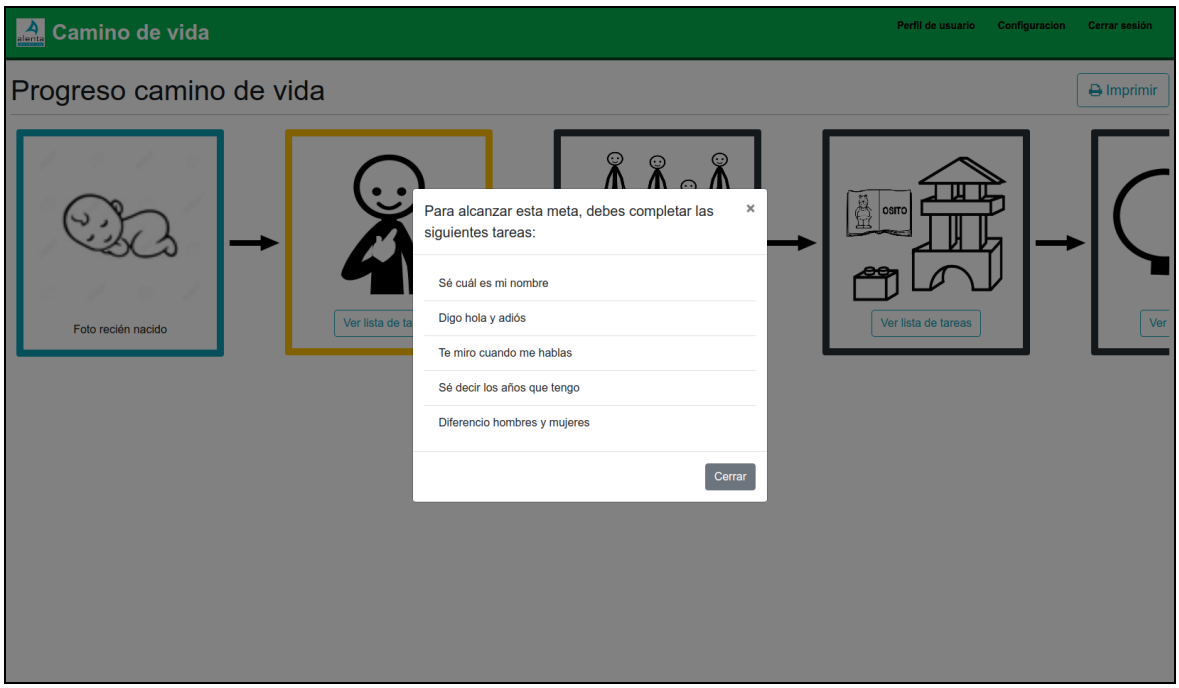


**Figura 0-21: Captura de pantalla con la lista de tareas para completar una meta**

## *C. Manual de instalación*

Antes de empezar con los pasos para realizar el despliegue de la aplicación, mencionar que este manual está orientado para la instalación en el sistema operativo Ubuntu 18.04, que es el que tiene instalado el portátil que se ha usado como servidor de la aplicación. Sin más dilación pasamos a los pasos que hay que seguir para realizar la instalación:

En primer lugar, tenemos que instalar el sistema gestor de base de datos a usar, en nuestro caso es PostgreSQL. Para realizar la instalación usamos el comando en la terminal:

```
sudo apt-get install postgresql postgresql-contrib
```

Una vez ha finalizado la instalación procedemos a crear el usuario que va a usar el *framework* de Django para autenticarse y poder realizar los cambios que sean necesarios en la base de datos. En este punto también crearemos la base de datos que va a usar la aplicación. Para estas tareas usaremos los siguientes comandos:

```
sudo -u postgres createuser –interactive
sudo -u postgres createdb caminodevidaDB
```

El siguiente paso consiste en preparar el entorno virtual con los paquetes que usará Django. Empezamos instalando la herramienta *virtualenv* para poder crear el entorno virtual, la instalación se realiza con el siguiente comando:

```
sudo pip3 install virtualenv
```

Ahora procedemos a crear el entorno virtual y a instalar las dependencias del proyecto, las cuales se comentaron en el capítulo de desarrollo, para ello utilizamos los siguientes comandos:

```
python3 -m venv myprojectenv
source myprojectenv/bin/activate
pip3 install Django==2.1.4
pip3 install psycopg2==2.7.6.1
pip3 install Pillow==6.0.0
```

La siguiente etapa consiste instalar el servidor HTTP Apache, en el cual se va a desplegar la aplicación web desarrollada, además del módulo *wsgi*, que es necesario para que el servidor pueda alojar aplicaciones desarrolladas en Python. Para realizarlo, utilizamos el siguiente comando en la terminal:

```
sudo apt-get install python3-pip apache2 libapache2-mod-wsgi-py3
```

Tras finalizar la instalación, procedemos a modificar el fichero `000-default.conf`, en donde especifícanos la configuración del servidor (se han omitido los comentarios y configuración que incluye apache por defecto):

```
<VirtualHost *:80>

    Alias /static /home/equipo/Projects/caminodevida/static
    <Directory /home/equipo/Projects/caminodevida/static>
        Require all granted
    </Directory>

    Alias /media /home/equipo/Projects/caminodevida/media
    <Directory /home/equipo/Projects/caminodevida/media>
        Require all granted
    </Directory>

    <Directory /home/equipo/Projects/caminodevida/CaminoDeVida>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    WSGIDaemonProcess CaminoDeVida python-
path=/home/equipo/Projects/caminodevida python-
home=/home/equipo/Projects/caminodevida/myprojectenv
    WSGIProcessGroup CaminoDeVida
    WSGIScriptAlias /
/home/equipo/Projects/caminodevida/CaminoDeVida/wsgi.py

    ...
</VirtualHost>
```

En penúltimo lugar, necesitamos realizar unos pequeños cambios en el fichero de configuración *settings.py* del proyecto de Django. Como por ejemplo incluir la variable `STATIC_ROOT = os.path.join(BASE_DIR, "static/")` para indicar la ruta donde se van a ubicar los ficheros estáticos que va a servir Apache.

Para terminar, ejecutamos el comando `python manage.py collectstatic`, necesario para copiar todos los ficheros estáticos a un mismo directorio, el cual se especificó anteriormente.